J-Link 用户指南

（本手册适用于 V6.0、V7.0、V8.0 版本的 J-LINK）

1. J-Link ARM JTAG 仿真器简介

J-Link 是 SEGGER 公司为支持仿真 ARM 内核芯片推出的 JTAG 仿真器。配合 IAR EWARM，ADS，KEIL，WINARM，RealView 等集成开发环境支持所有 ARM7/ARM9 内核芯片的仿真，通过 RDI 接口和各集成开发环境无缝连接，操作方便、连接方便、简单易学，是学习开发 ARM 最好最实用的开发工具。

J-Link ARM 主要特点

* IAR EWARM 集成开发环境无缝连接的 JTAG 仿真器
* 支持所有 ARM7/ARM9 内核的芯片，以及 cortex M3，包括 Thumb 模式
* 支持 ADS,IAR,KEIL,WINARM,REALVIEW 等几乎所有的开发环境
* 下载速度高达 ARM7:600kB/s，ARM9:550kB/s，通过 DCC 最高可达 800 kB/s
* 最高 JTAG 速度 12 MHz
* 目标板电压范围 1.2V –3.3V，兼容 5V
* 自动速度识别功能
* 监测所有 JTAG 信号和目标板电压
* 完全即插即用
* 使用 USB 电源（但不对目标板供电）
* 带 USB 连接线和 20 芯扁平电缆
* 支持多 JTAG 器件串行连接
* 标准 20 芯 JTAG 仿真插头
* 选配 14 芯 JTAG 仿真插头
* 带 J-Link TCP/IP server，允许通过 TCP/ IP 网络使用 J-Link

J-Link 支持 ARM 内核

* ARM7TDMI（Rev 1）
* ARM7TDMI（Rev 3）
* ARM7TDMI-S（Rev 4）
* ARM720T * ARM920T
*ARM926EJ-S
*ARM946E-S
*ARM966E-S
* ARM11
* Cortex-M3

速度信息

Revision	ARM7 Memory download	ARM9 Memory download
J-Link Rev. 1-4	150.0 kB/s (4MHz JTAG)	75.0 kB/s (4MHz JTAG)
J-Link Rev. 5-8	720.0 kB/s (12MHz JTAG)	550.0 kB/s (12MHz JTAG)
J-Trace Rev. 1	420.0 kB/s (12MHz JTAG)	280.0 kB/s (12MHz JTAG

2. J-LINK 驱动安装

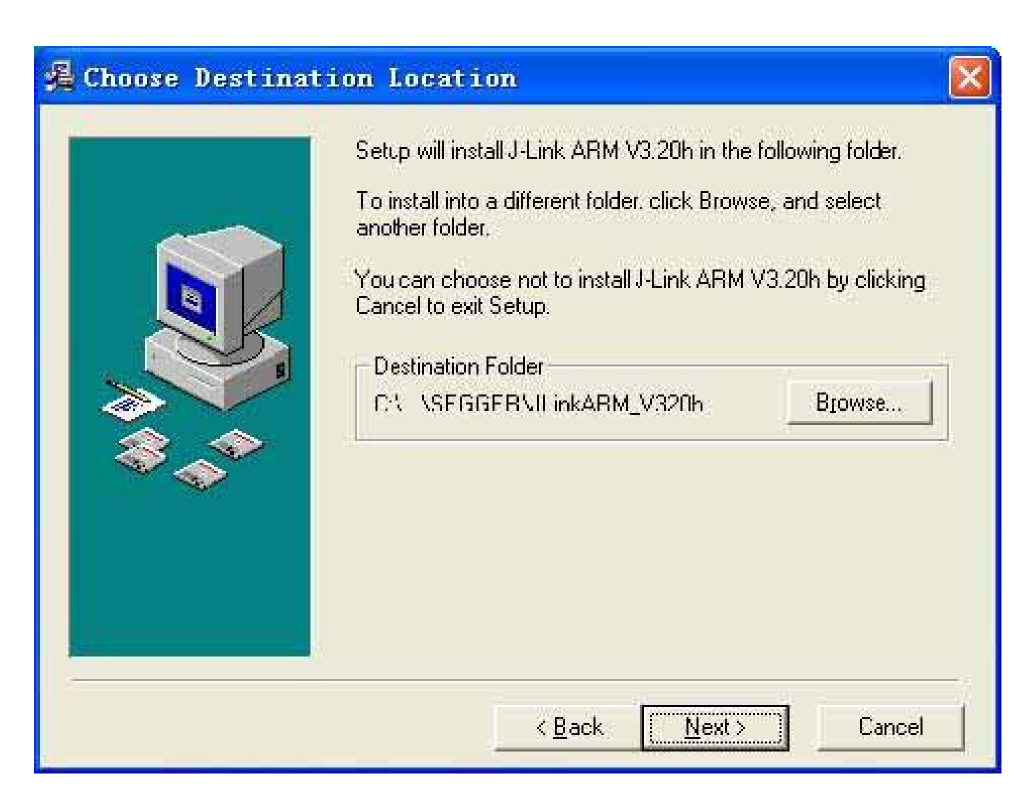
J-LINK 是本站开发的兼容产品，具有一样的性能，但是却只有十分之一的价格！

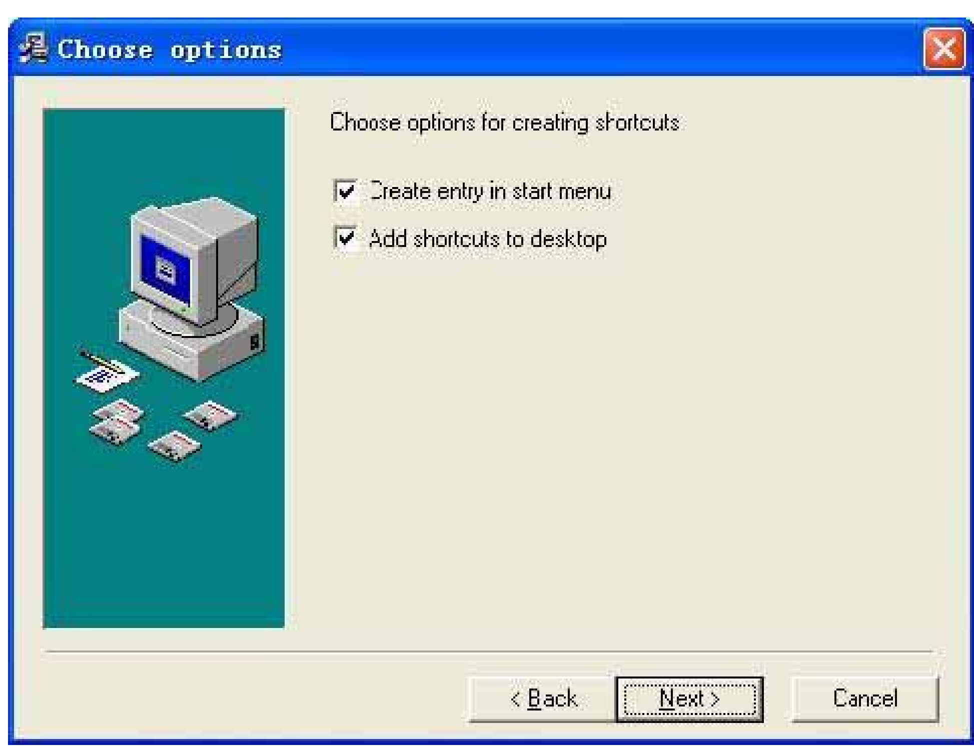
首先到 http://www.segger.com/download_jlink.html 下载最新的 J-LINK驱动软，J-Link ARM software and documentation pack ，内含USB driver, J-Mem, J-Link.exe and DLL for ARM, J-Flash and J-Link RDI。

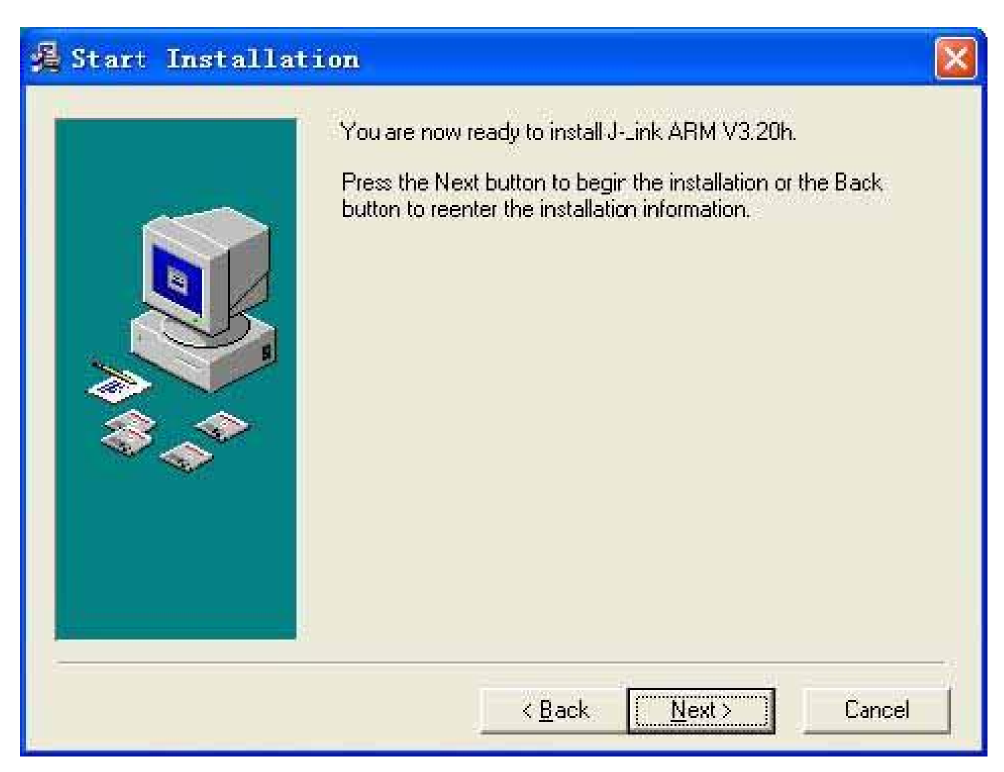
注意：SEGGER 公司升级比较频繁，请密切留意 SEGGER 公司网站，下载最新驱动，以支持更多器件！

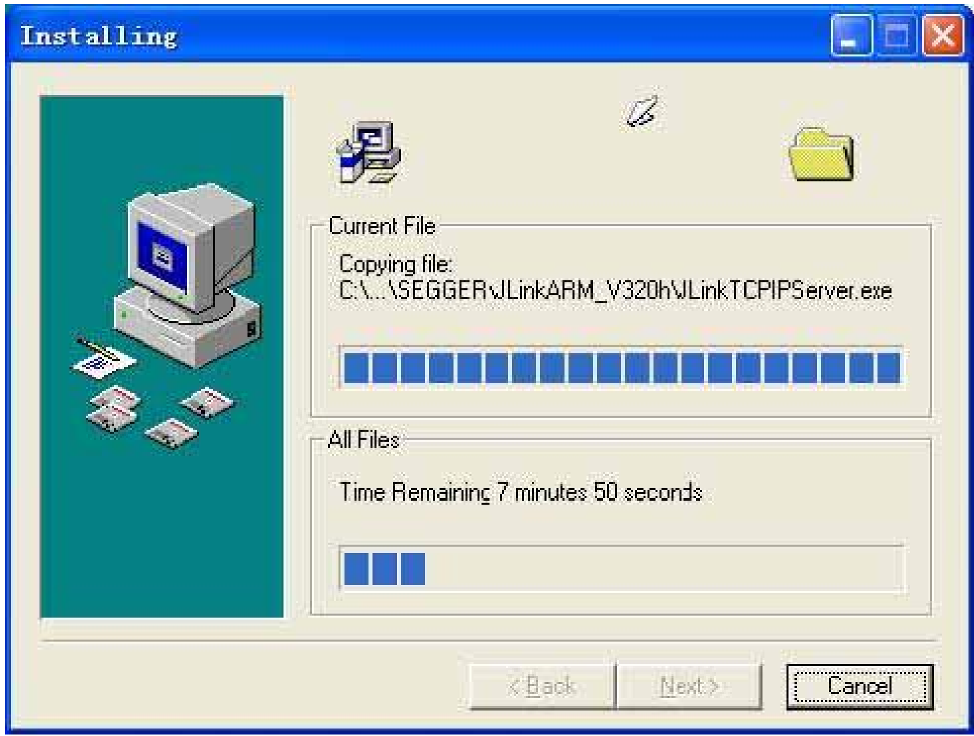
安装驱动很简单，只要将下载的 ZIP 包解压，然后直接安装即可，默认安装，一路点击“NEXT”即可：




Choose Destination Location
Setup will install J-Link ARM V3.20h in the following folder.
To install into a different folder, click Browse, and select another folder.
You can choose not to install J-Link ARM V3.20h by clicking Cancel to exit Setup.
Destination Folder
Browse...
< Back
Next >
Cancel


Choose options
Choose options for creating shortcuts
Add shortcuts to desktop
< Back
Next >
Cancel

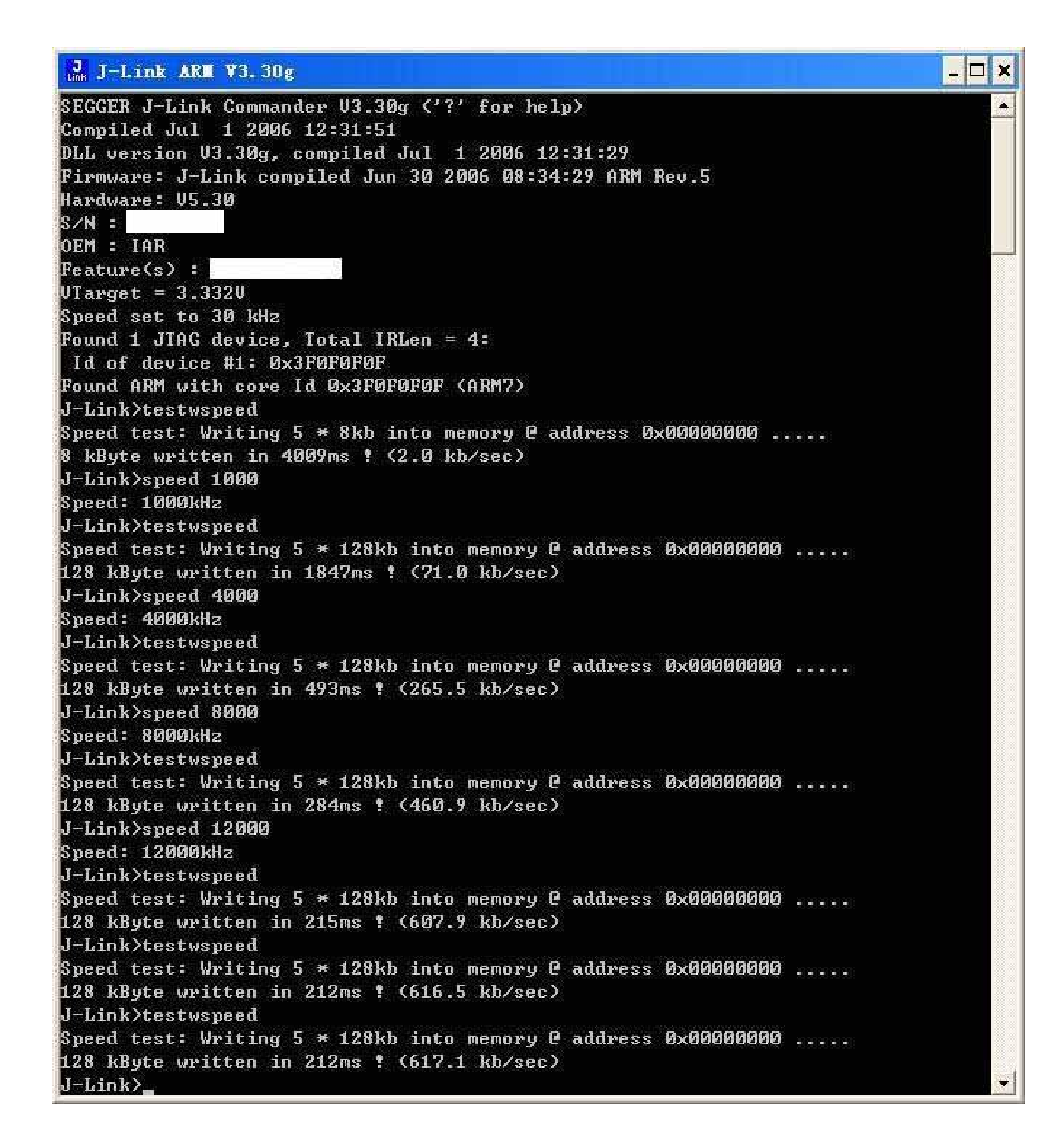

Start Installation
You are now ready to install J-Link ARM V3.20h.
Press the Next button to begin the installation or the Back button to reenter the installation information.
< Back
Next >
Cancel


Installing
Current File
Copying file:
C:\...\SEGGER\JLinkARM_V320h\JLinkTCPIPServer.exe
All Files
Time Remaining 7 minutes 50 seconds
< Back
Next >
Cancel



安装完成后， 请插入 JLINK 硬件，然后系统提示发现新硬件，一般情况下会自动安装驱动，如果没有自动安装，请选择手动指定驱动程序位置（安装目录），然后将驱动程序位置指向到 JLINK 驱动软件的安装目录下的 Driver 文件夹，驱动程序就在改文件夹下。

安装完成可以桌面出现两个快捷图标，J-Link ARM 可以用来进行设置和测试，下面我们看一下 J-LINK 的测试数据（在 7X256 EK 上测试）：



注意：由于 ARM7TDMI-S 内核的特殊性，LPC2000 系列的 JTAG

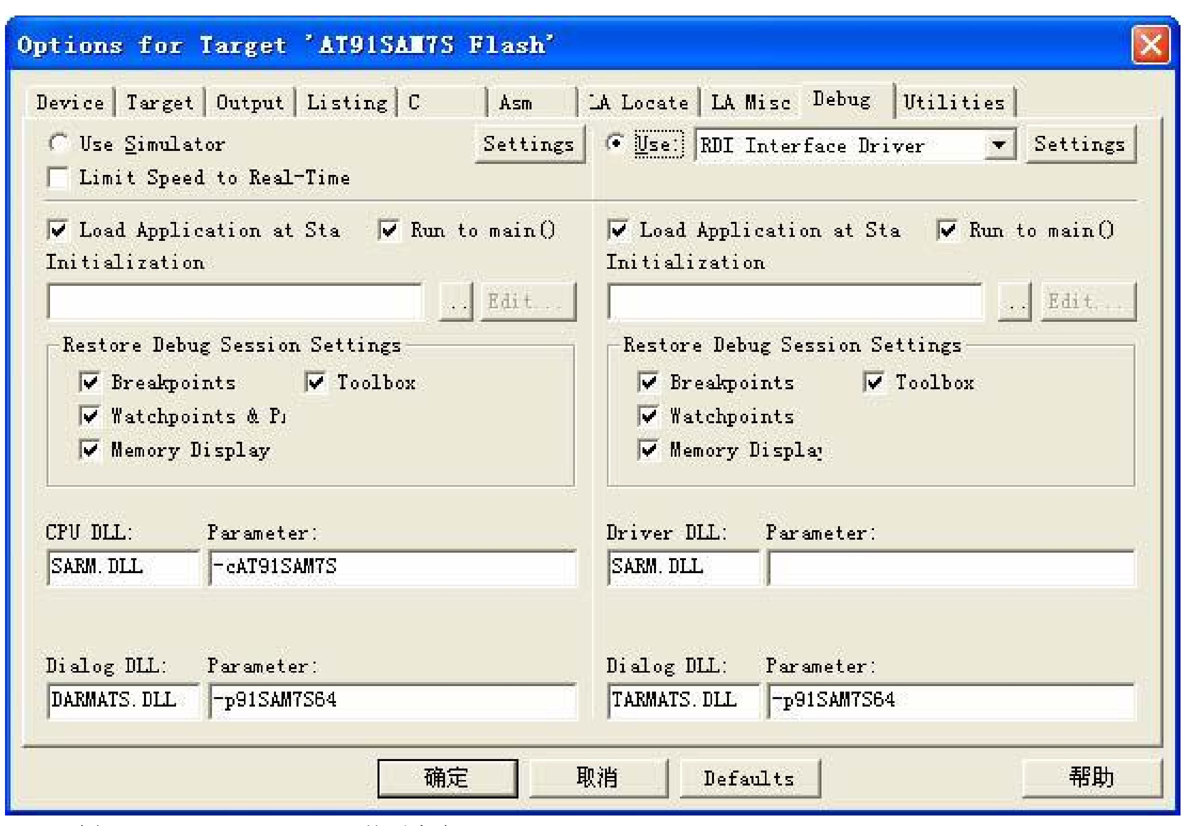
速度最高只能达到 1/6 系统时钟，一般最高是 4.8M，如果 JTAG 速度超过 4.8M，J-LINK ARM 将提示找不到 LPC2000。这是 LPC2000 内核的局限，与 JLINK 无关！同时在开发环境下调试 LPC2000 的时候，也注意 JTAG 时钟不能设置超过 4.8M，不然将工作不稳定，甚至无法连接到目标芯片。其他芯片，如 ATMEL 的 SAM7 系列无此问题。

注意：由于 J-LINK 版本不断更新，该测试数据也会有相应变化，这里提供的测试数据主要是为了验证我们的产品和原装产品的性能差异。

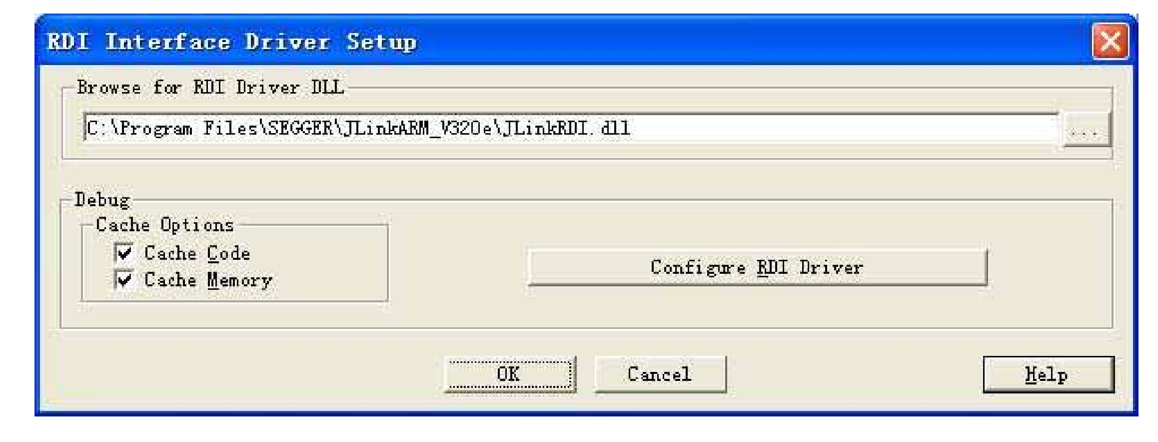
3. J-LINK（JLINK）在各个主流开发环境下的设置

3.1. Keil开发环境

下面简单叙述一下在 Keil 下如何使用 J-Link：

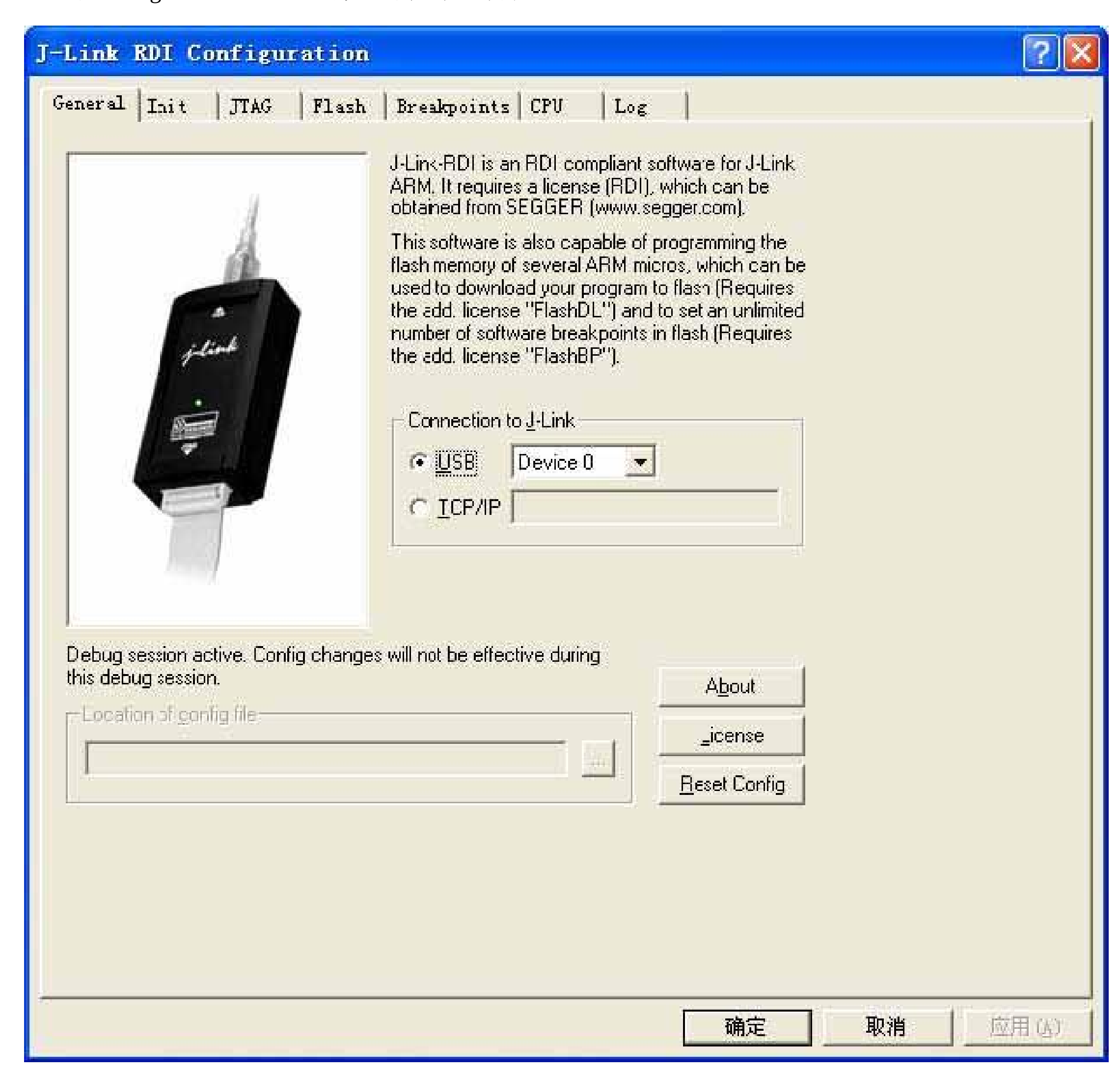


选择“RDI Interface Driver”，然后点击“Settings”：

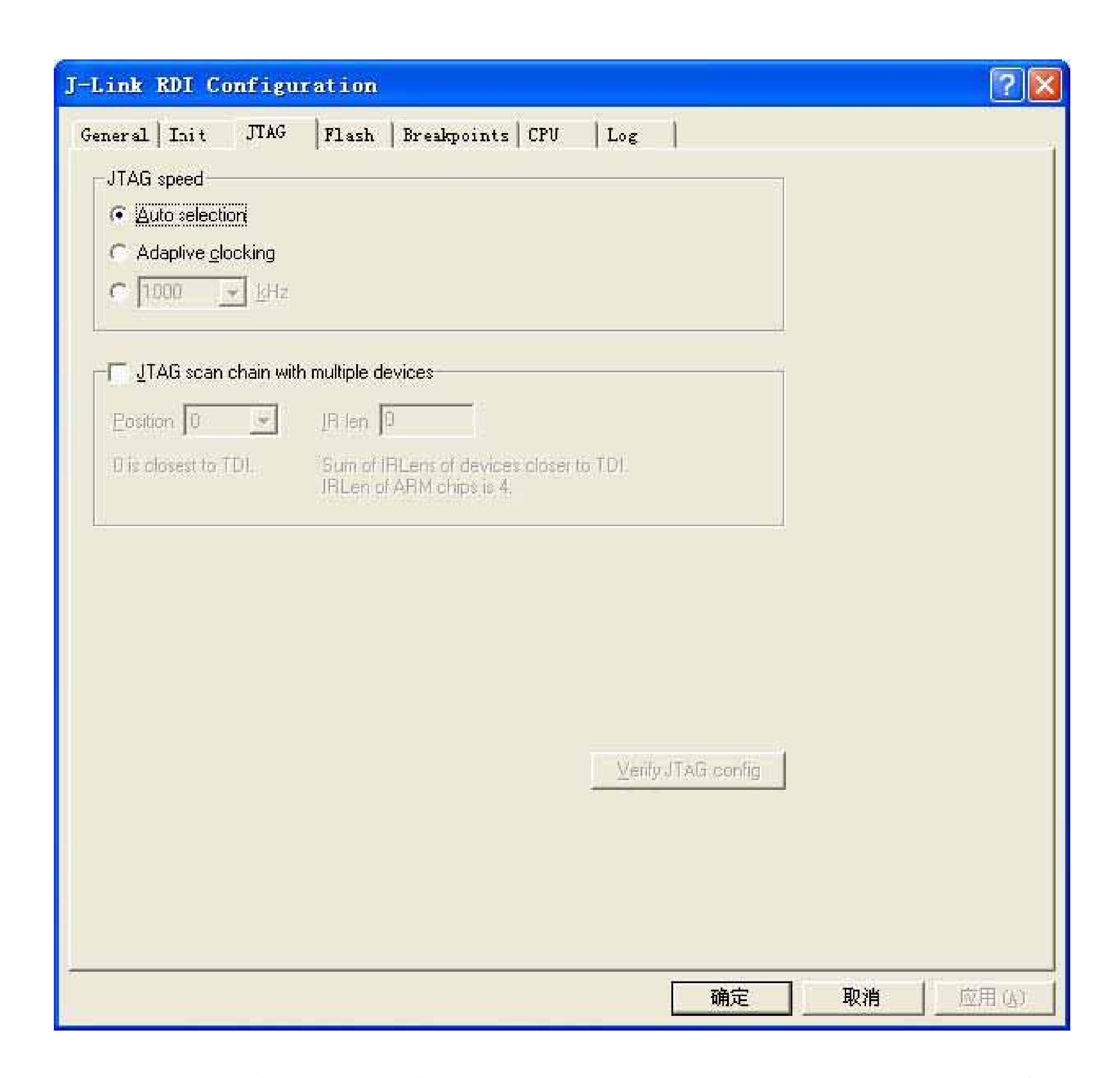


请点击“…”，指向到 JLINK 安装目录。

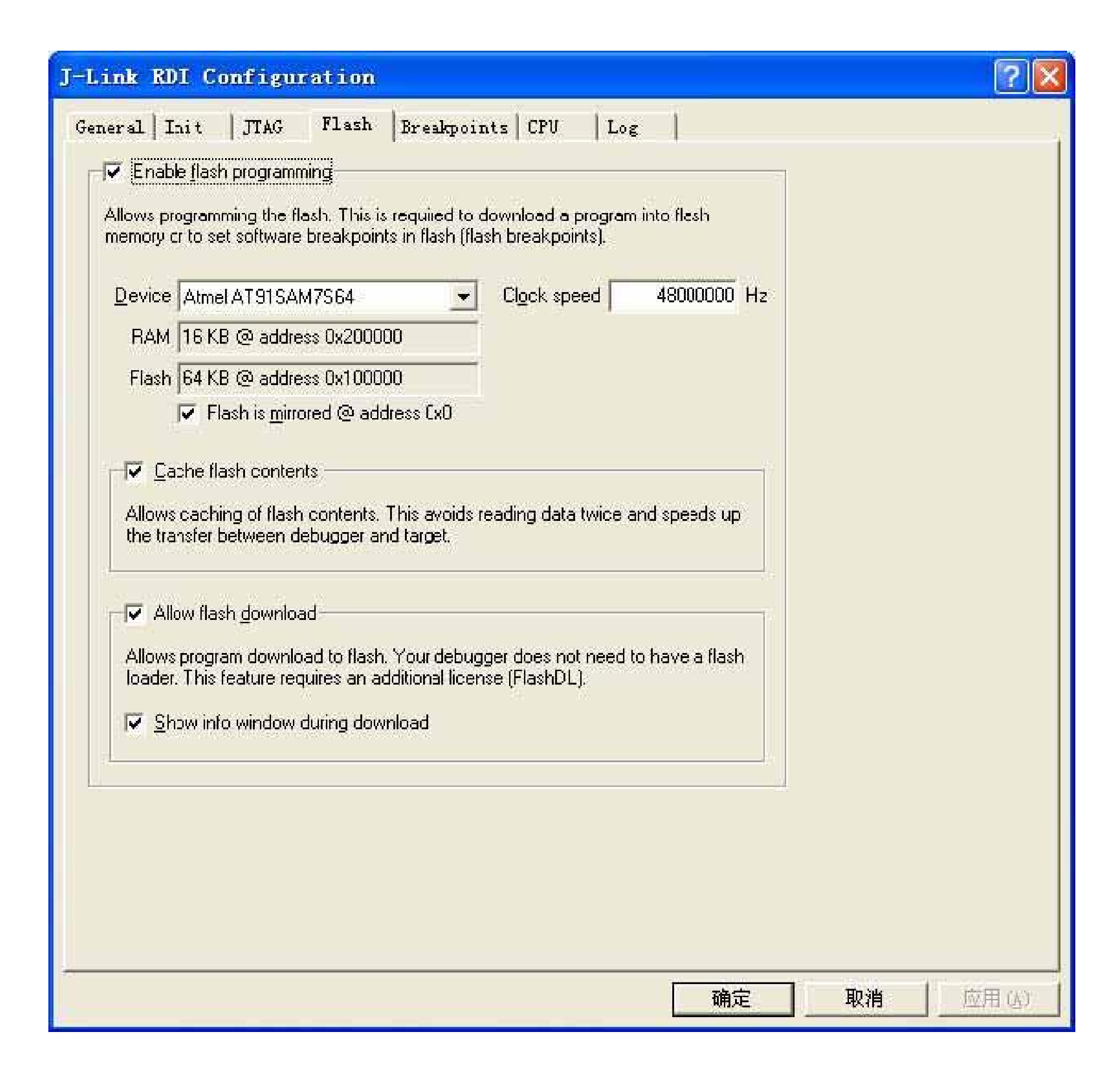
点击“Configure RDI Driver”出现以下几个选项卡：



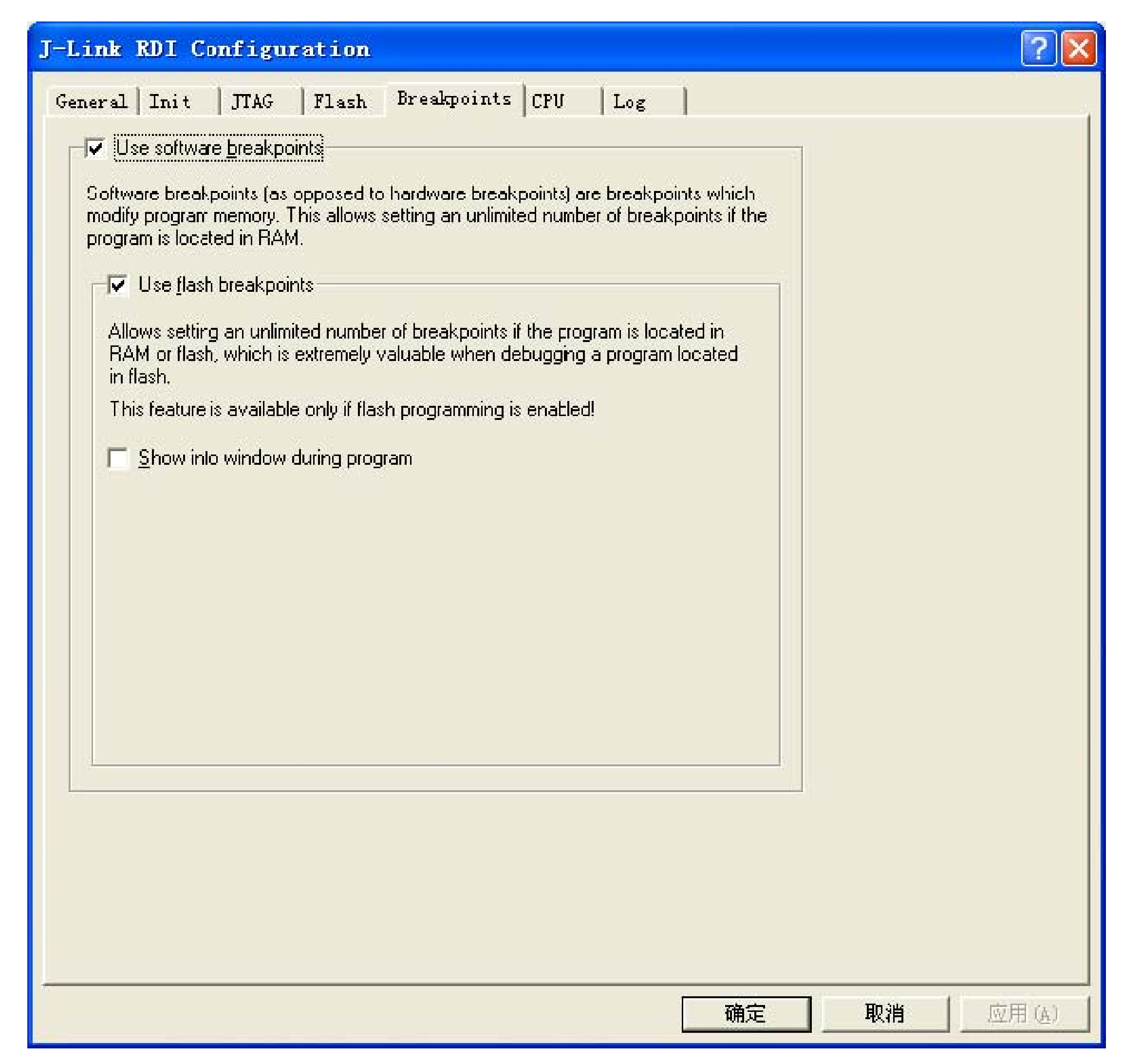
如果是本机调试，直接使用 USB 口即可；如果是在局域网内调试，可以选择 TCP/IP，然后指定一个挂接了 J-LINK 的 PC 的 IP 地址。



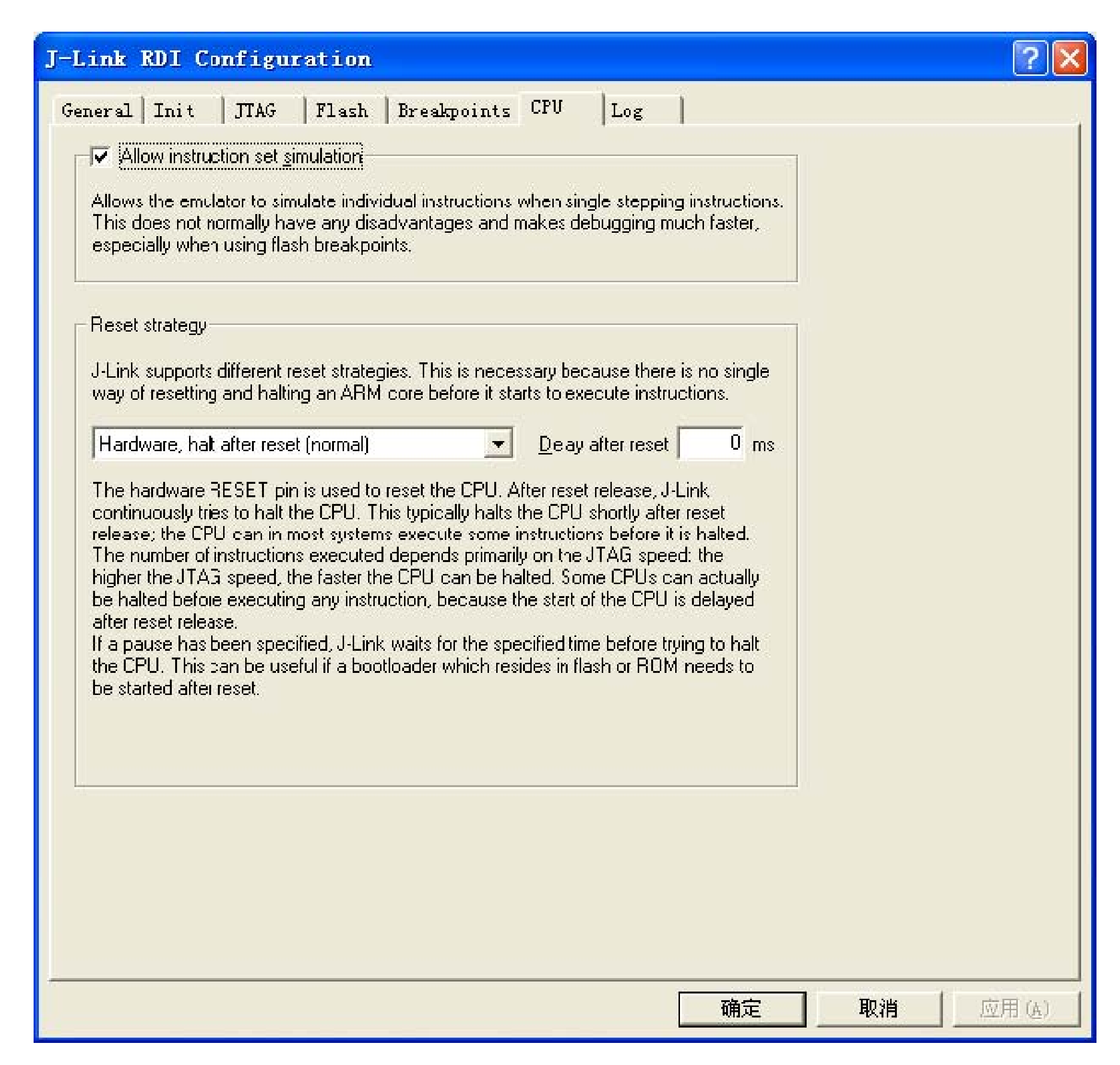
设置 JTAG 速度，如果是-S 内核，建议使用 Auto 方式，如果是非-S 内核，可以直接使用最高速度 12M。使用过程中如果出现不稳定情况，可以将 JTAG 时钟速度适当调低。



使能 FLASH 编程功能，如果你的目标芯片是带片内 FLASH 的 ARM，就可以使用该功能，这样子在调试前 J-LINK 就会先编程 FLASH。



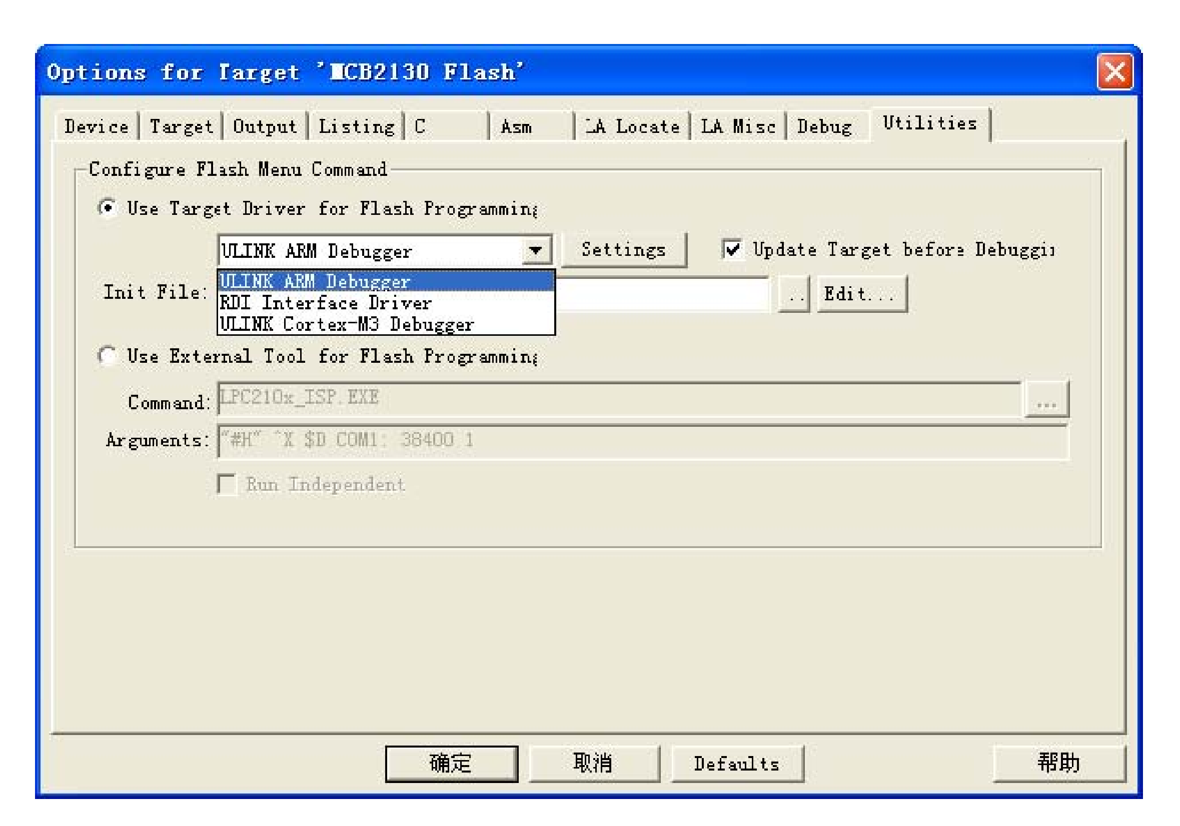
使用软件断点，如果是带片内 FLASH 的 ARM，建议使用该功能，可以打上 n 多断点，方便调试。



在这里可以设置 Reset 策略，有好几种 Reset 策略可选，同时可以设置 Reset 后的延迟时间，这个设置对于需要较长复位时间的芯片较为有用，如 AT91RM9200。

以上设置是用 JLINK 进行 Debug 的设置，如果要使用 KEIL 提供的 即“DOWNLOAD”功能则还需要在“Utilities”菜单里面进行和“Debug”一样的设置：

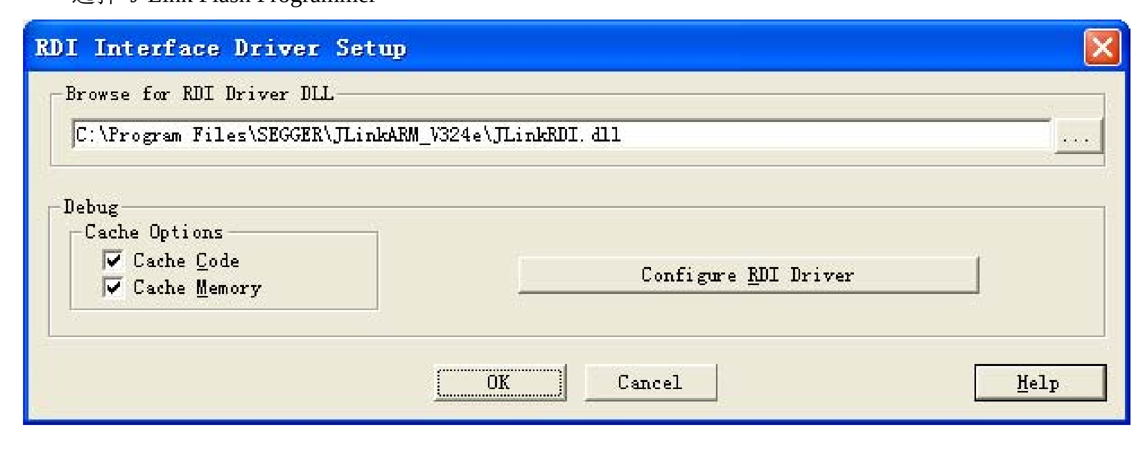
的 ARM7/ARM9 芯片。



选择“RDI Interface Driver”，然后点击“Settings”



选择“J-Link Flash Programmer”

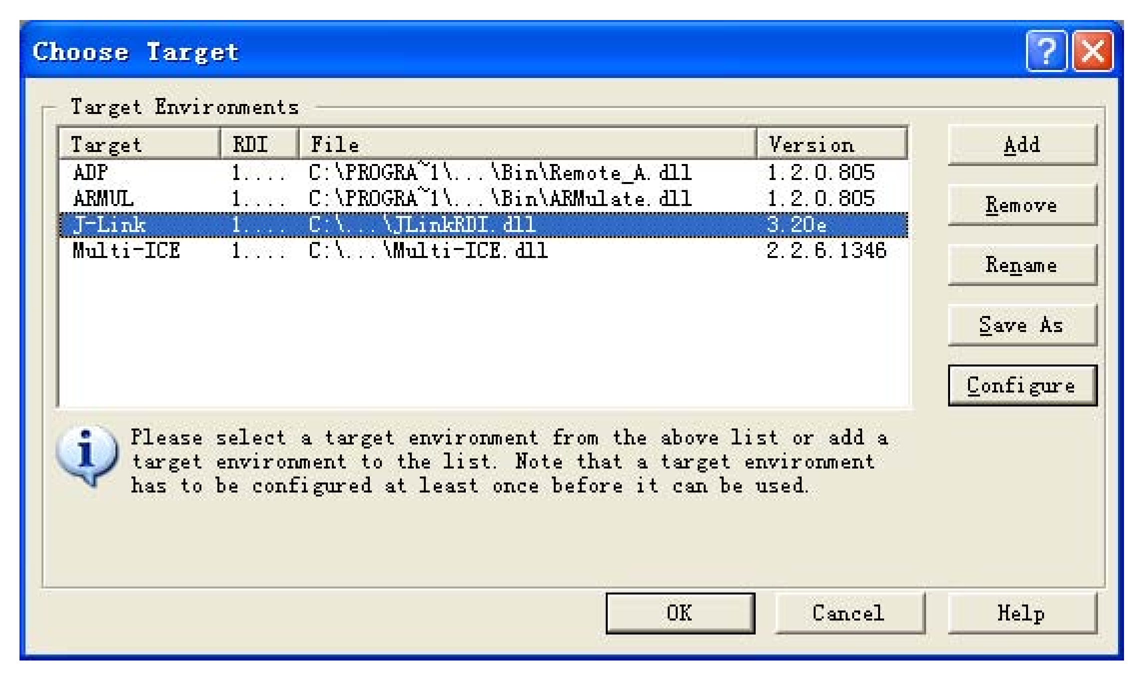


接下来的设置就同“Debug”下设置一样了。

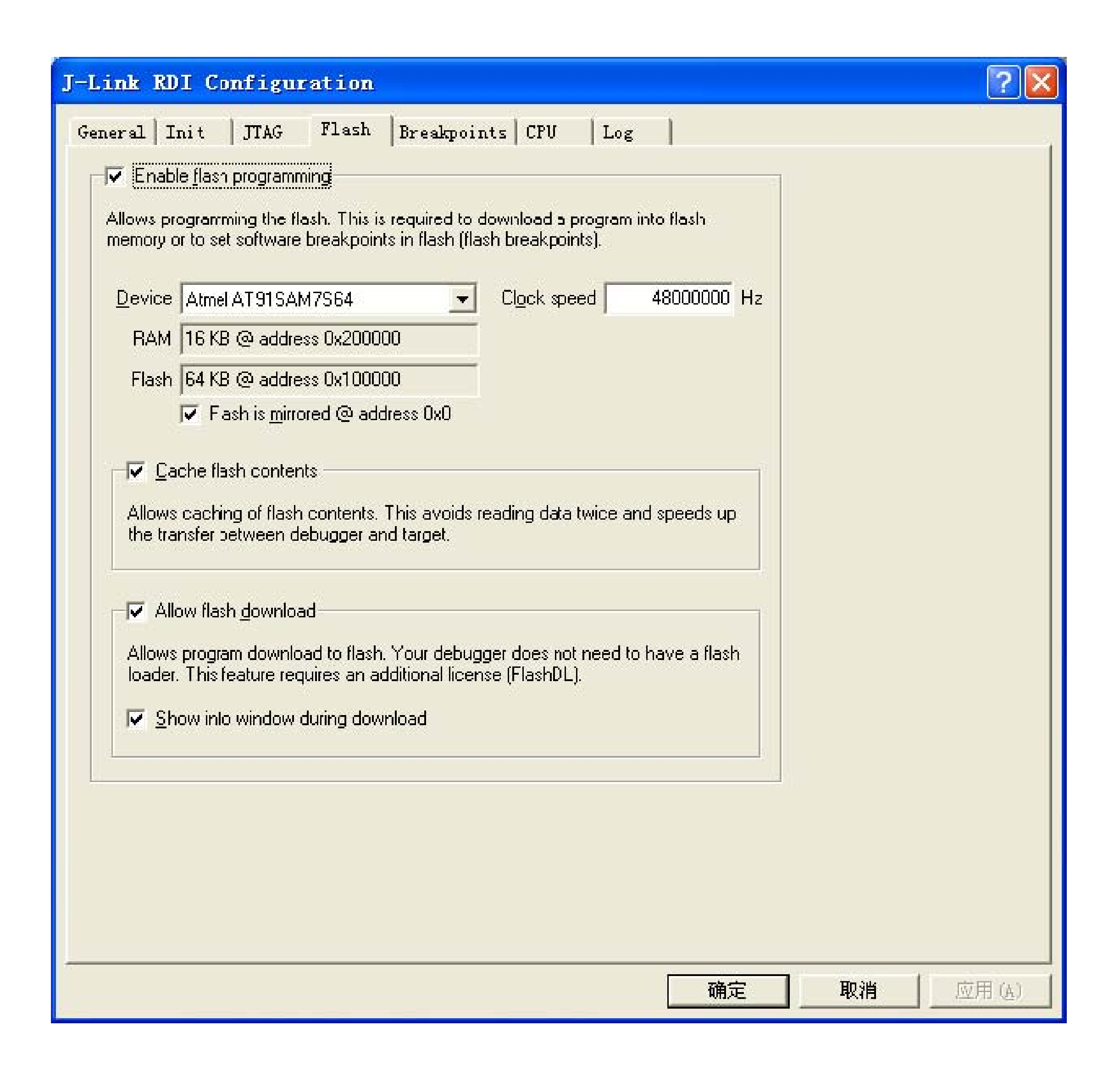
完成以上设置后，就可以通过“LOAD”按钮进行直接下载。 注意，该功能只支持具备片内 Flash 的 ARM7/ARM9 芯片。

3.2. ADS开发环境

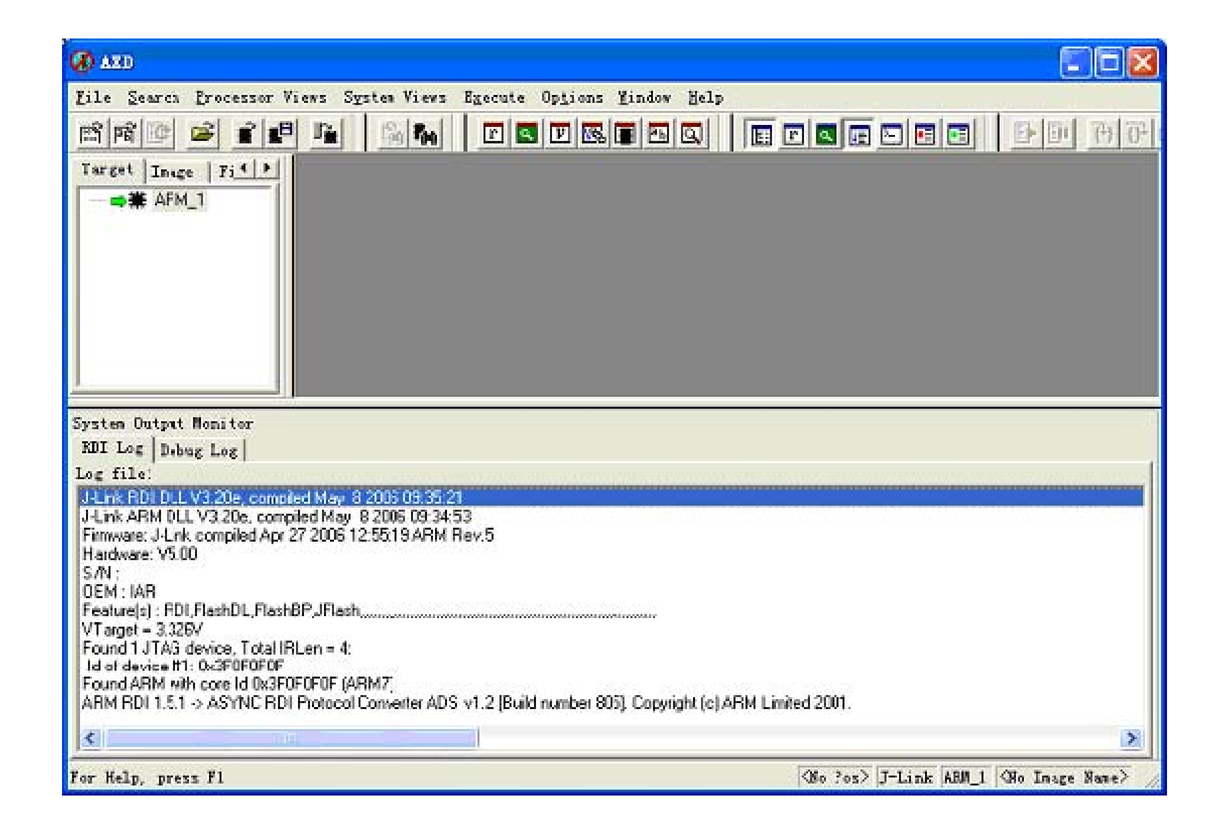
点击“Add”，选择 JLINKRDI.DLL：



点击“Configure”，出现以下内容



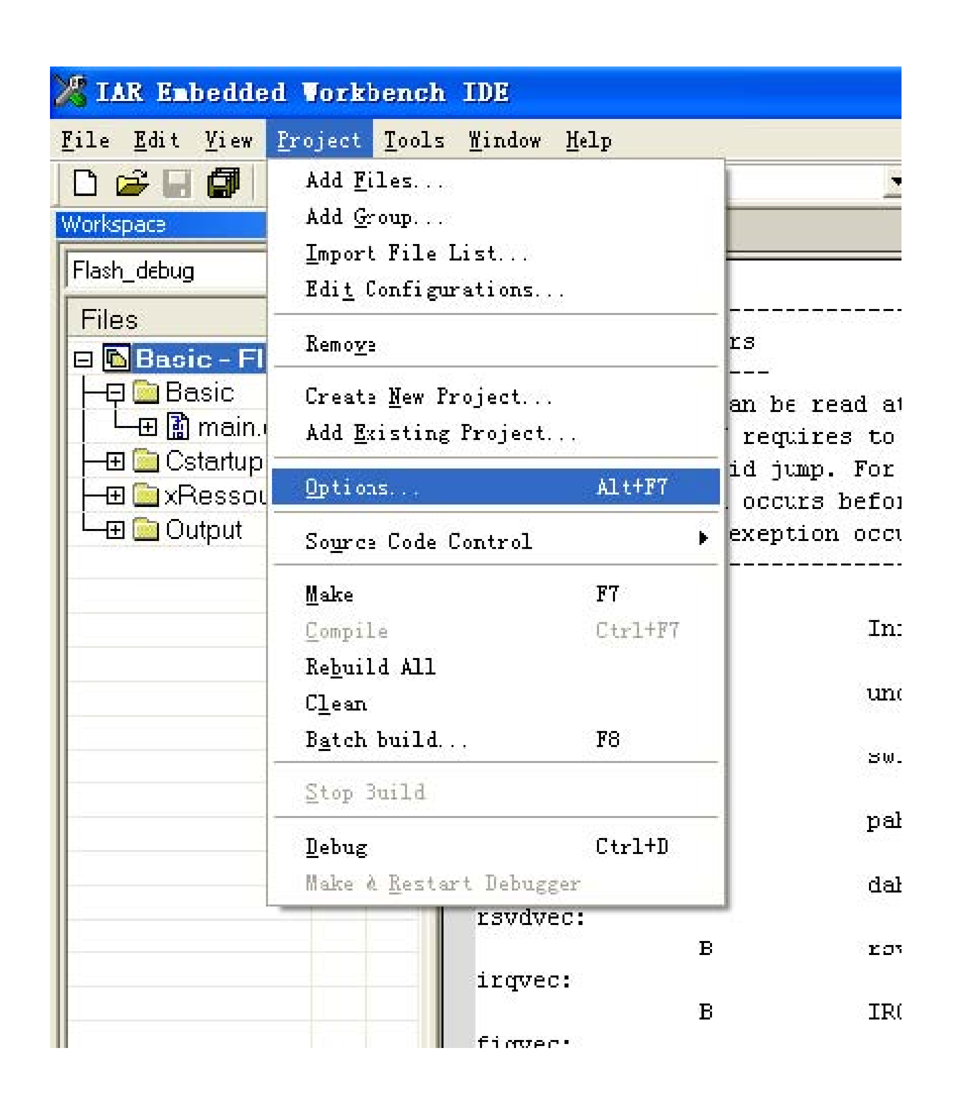
进入 AXD 后的信息(注意 LOG FILE 的内容):

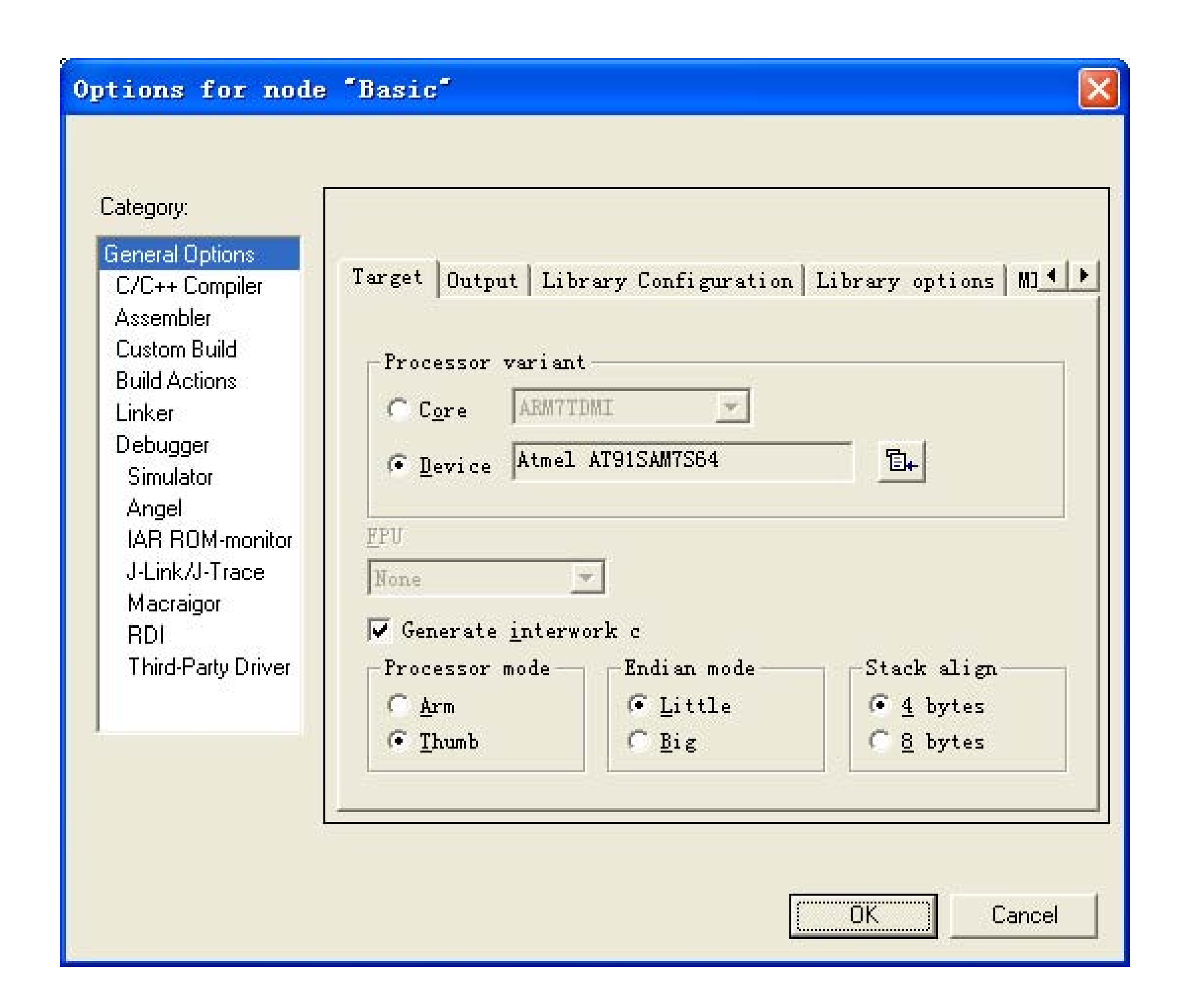


3.3.IAR开发环境

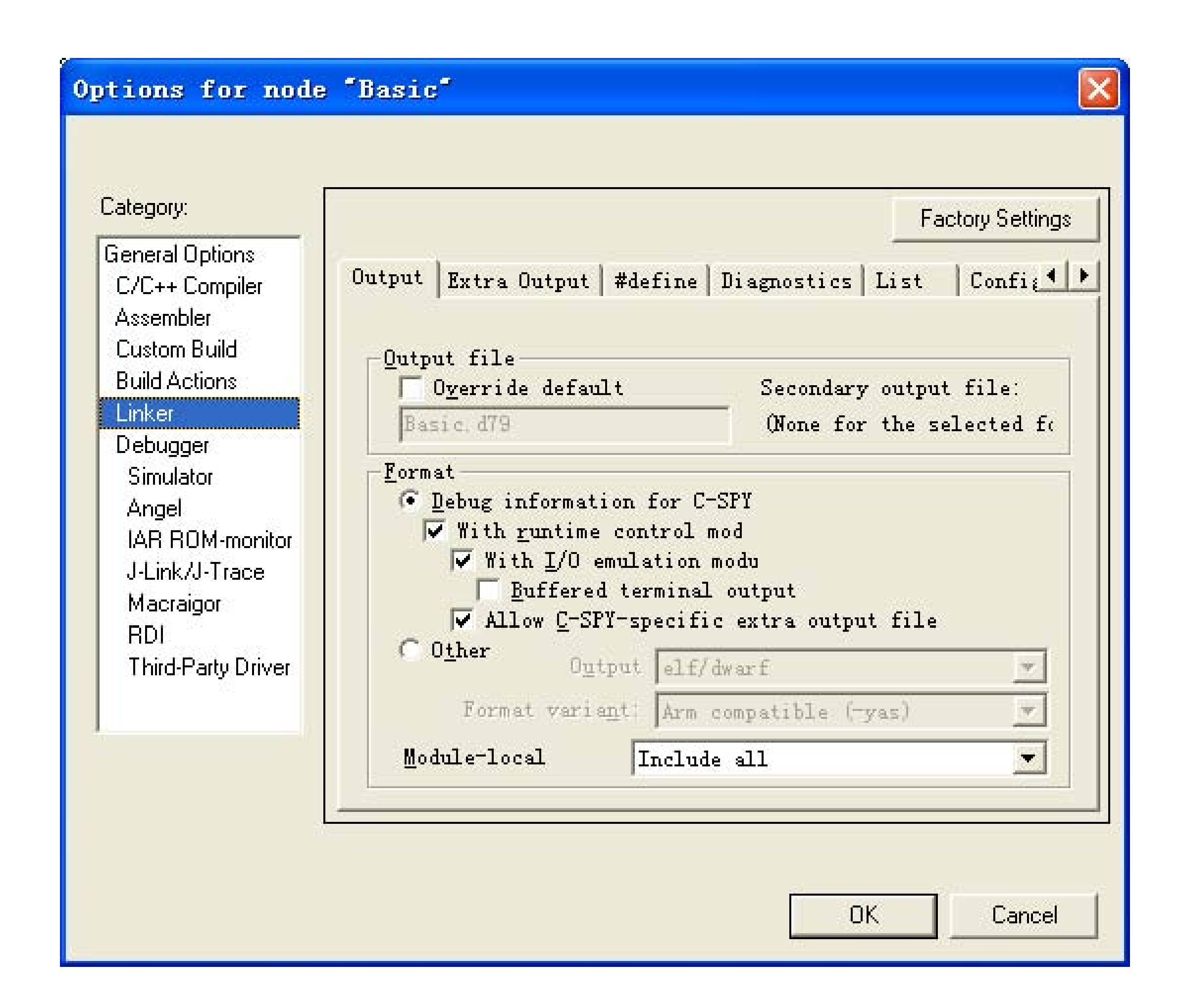
在 IAR 既可以使用 IAR 提供的 JLINK 的驱动，也可以使用 RDI 接口的驱动，推荐使用 RDI 接口的驱动，因为 IAR 版本的 JLINK 对速度和功能做了限制。

首先打开一个工程，然后按照下图开始进入设置页面：

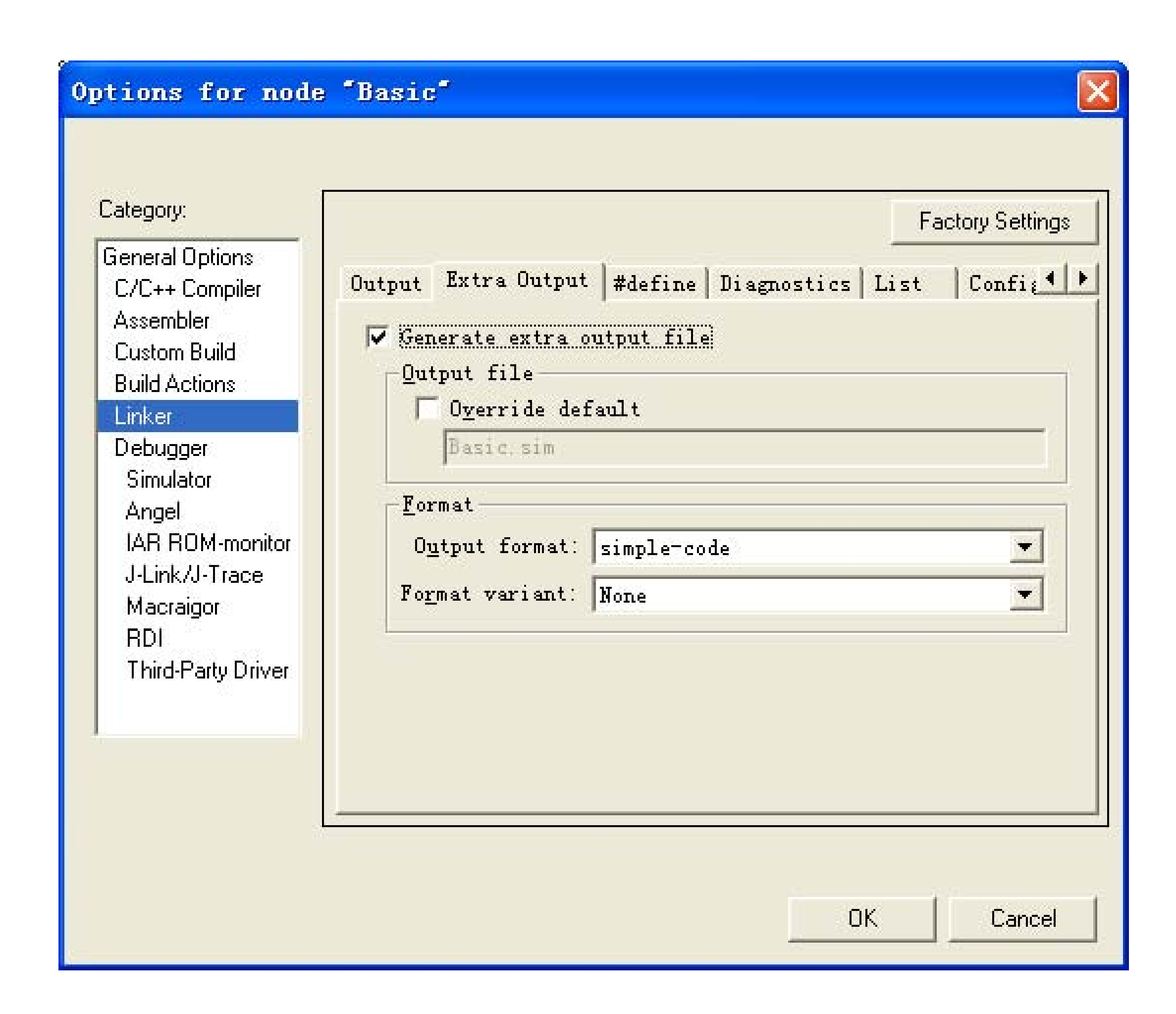

IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Flash_debug
Files
Basic - Fl
Basic
main.
Cstartup
xRessou
Output
Add Files...
Add Group...
Import File List...
Edit Configurations...
Remove
Create New Project...
Add Existing Project...
Options... Alt+F7
Source Code Control
Make F7
Compile Ctrl+F7
Rebuild All
Clean
Batch build... F8
Stop Build
Debug Ctrl+D
Make & Restart Debugger
rsvdvec:
irqvec:

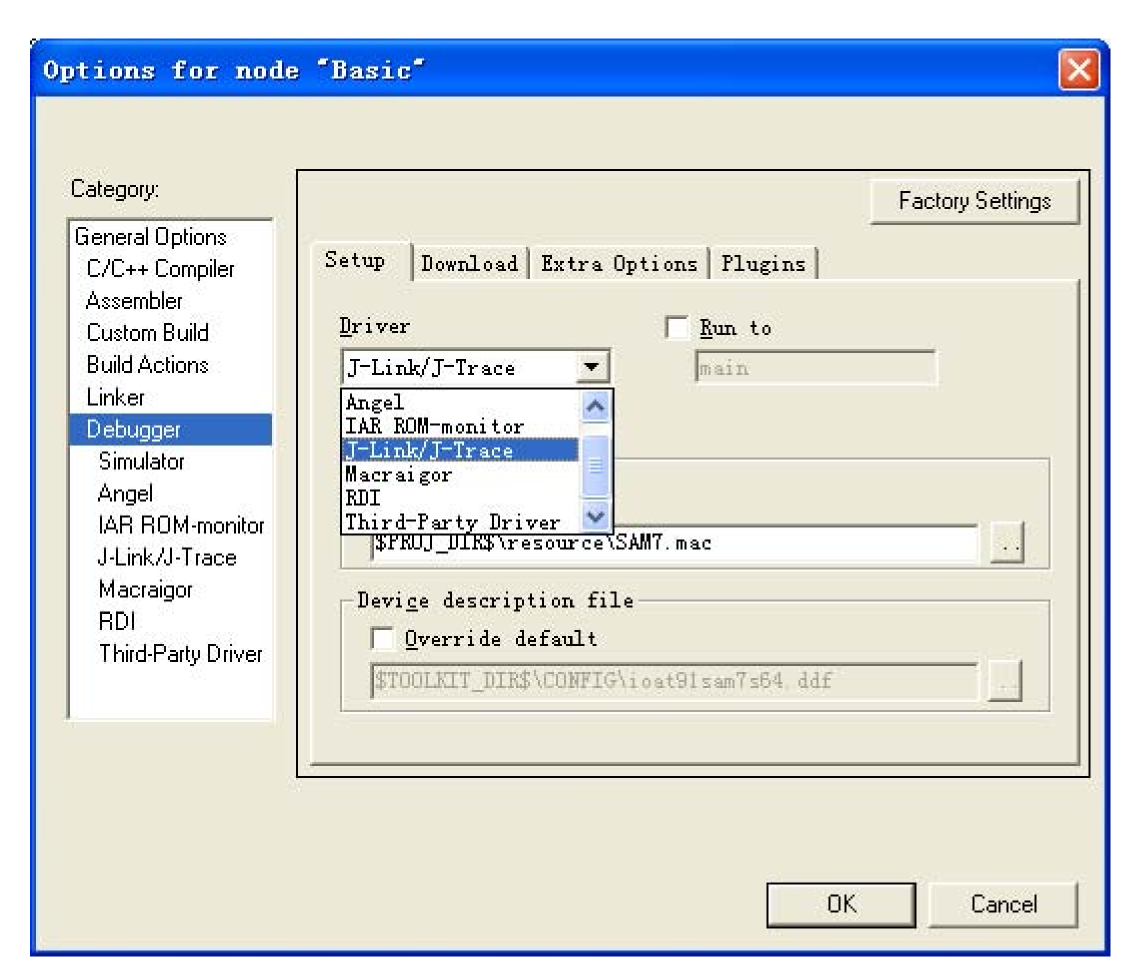


Options for node "Basic"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Simulator
Angel
IAR ROM-monitor
J-Link/J-Trace
Macraigor
RDI
Third-Party Driver
Target
Output
Library Configuration
Library options
Processor variant
Core
ARM7TDMI
Device
Atmel AT91SAM7S64
FPU
None
Generate interwork c
Processor mode
Arm
Thumb
Endian mode
Little
Big
Stack align
4 bytes
8 bytes
OK
Cancel

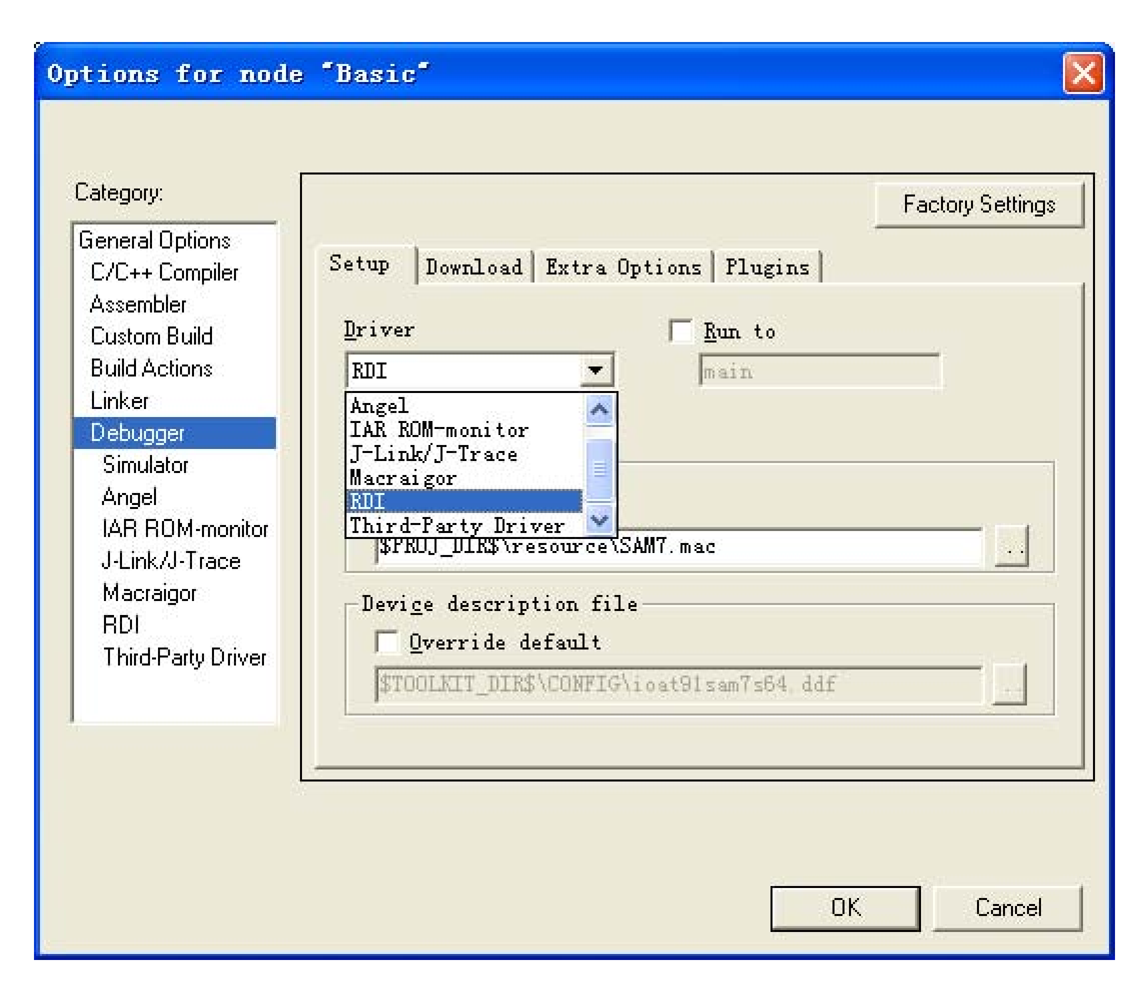


Options for node "Basic"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Simulator
Angel
IAR ROM-monitor
J-Link/J-Trace
Macraigor
RDI
Third-Party Driver
Factory Settings
Output
Extra Output
#define
Diagnostics
List
Output file
Override default
Basic.d79
Secondary output file:
Format
Debug information for C-SPY
With runtime control mod
With I/O emulation modu
Buffered terminal output
Allow C-SPY-specific extra output file
Other
Output
elf/dwarf
Format variant:
Arm compatible (-yas)
Module-local
Include all
OK
Cancel

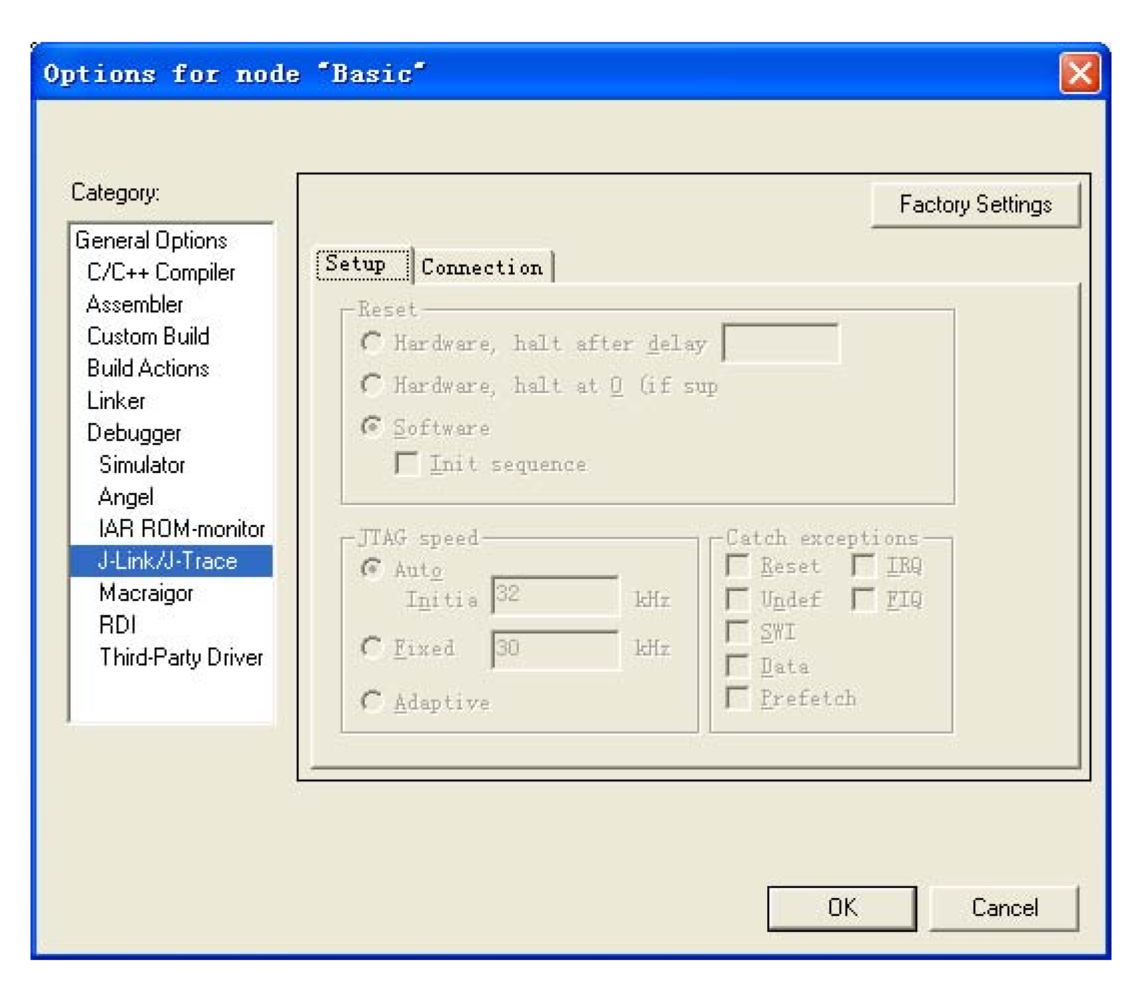

Options for node "Basic"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Simulator
Angel
IAR ROM-monitor
J-Link/J-Trace
Macraigor
RDI
Third-Party Driver
Factory Settings
Output
Extra Output
#define
Diagnostics
List
Generate extra output file
Output file
Override default
Basic.sim
Format
Output format:
simple-code
Format variant:
None
OK
Cancel



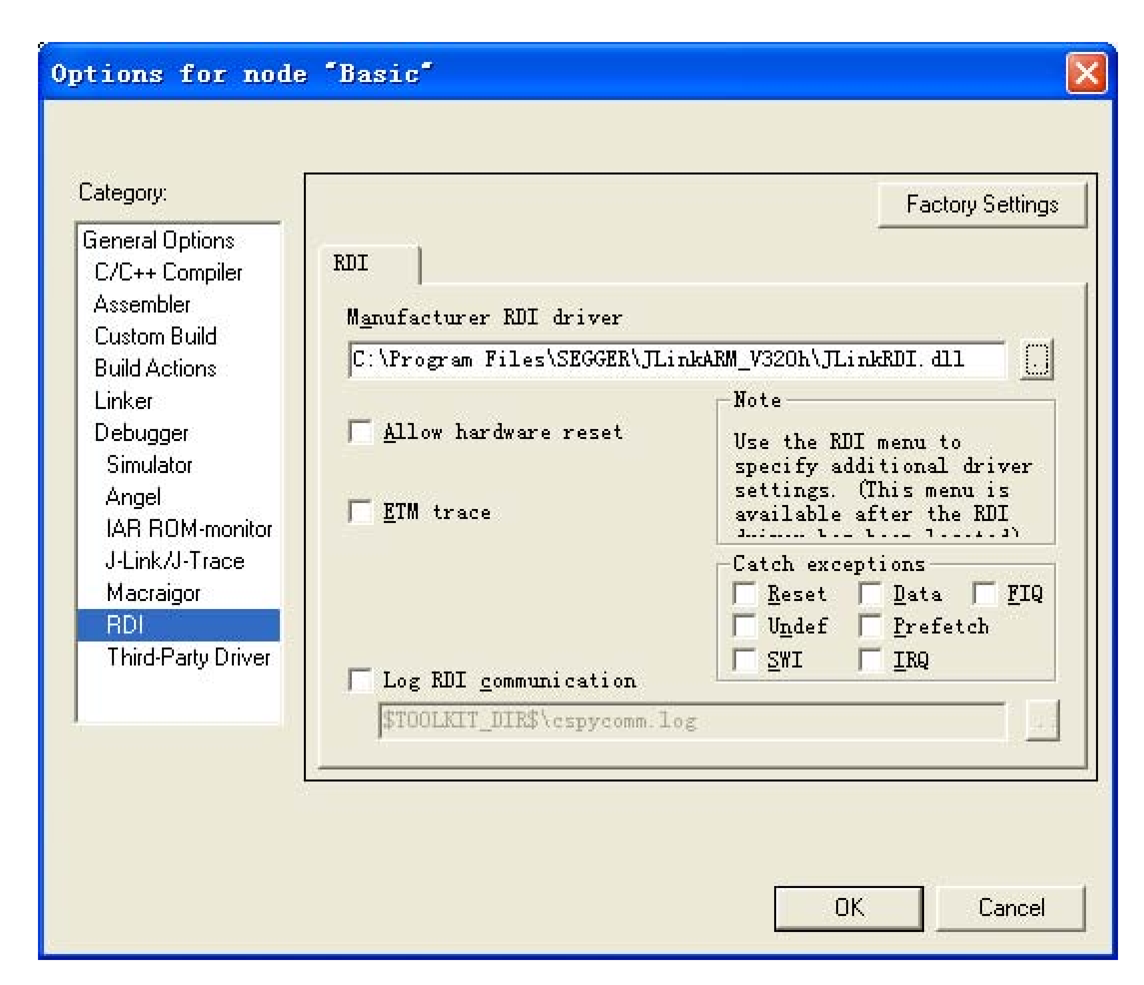
如果购买的是 IAR 版本的 JLINK，请选择“J-LINK/J-TRACE”；如果购买的是全功能版本 JLINK，则既可以选择“J-LINK/J-TRACE”，也可以选择“RDI”，建议选择“RDI”，以提升性能。



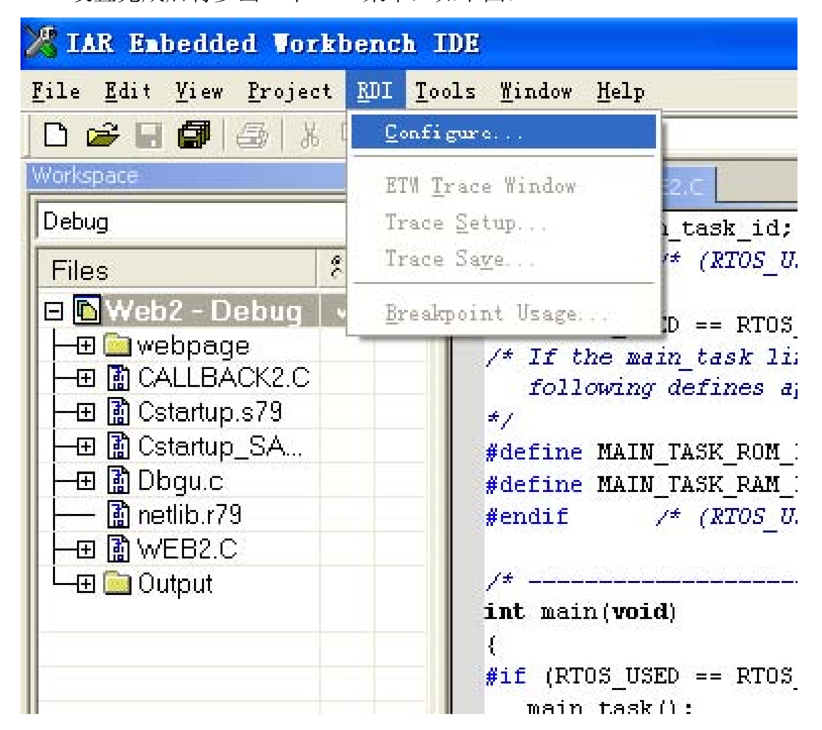
如果选择“J-LINK/J-TRACE”，则无需额外设置：



如果选择“RDI”，则还需要指定 JLINKRDI.DLL 的位置：



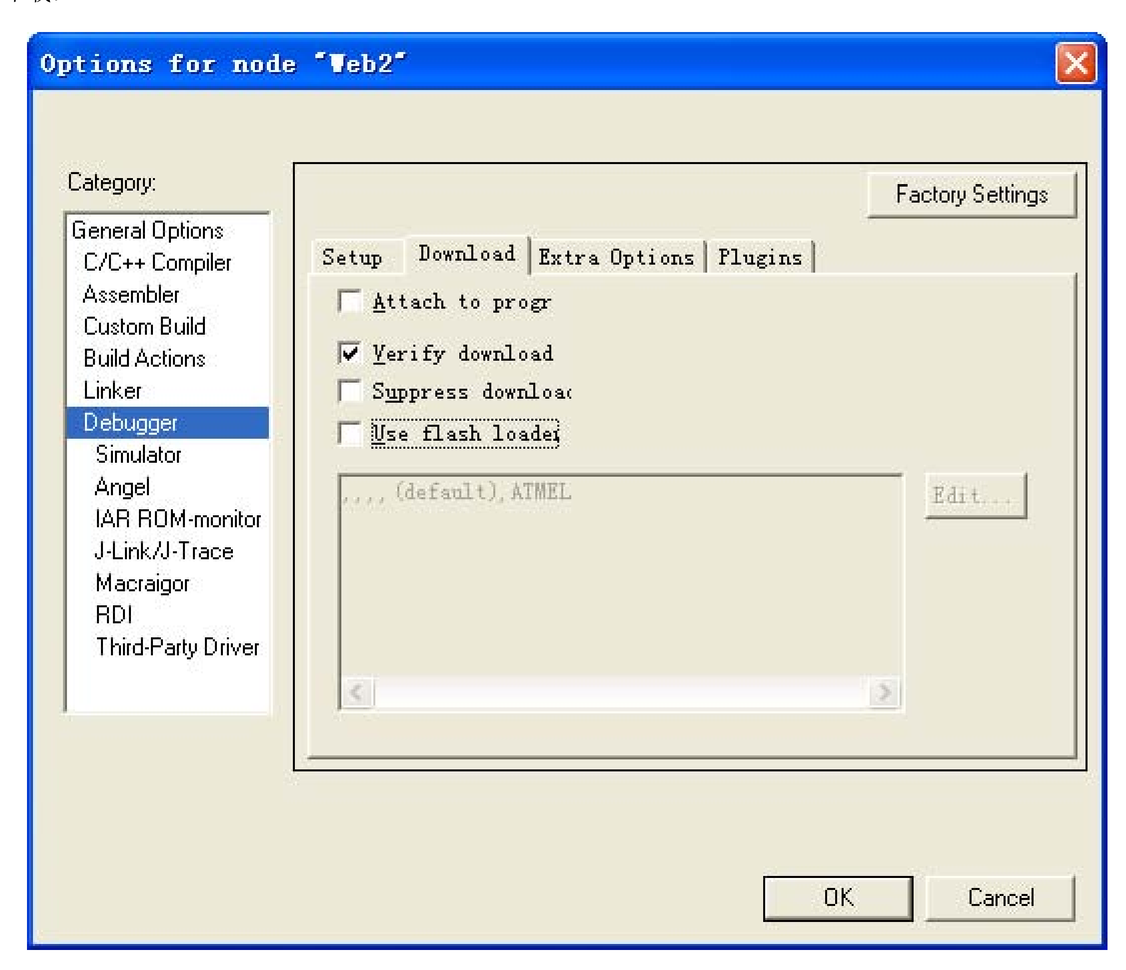
设置完成后将多出一个 RDI 菜单，如下图：



在 RDI 菜单下有“CONFIGURE”选项，这里可以对 JTAG 时钟，FLASH，断点，CPU 等进行设置，

请注意里面的 FLASH 和 CPU 型号与目标板相吻合。

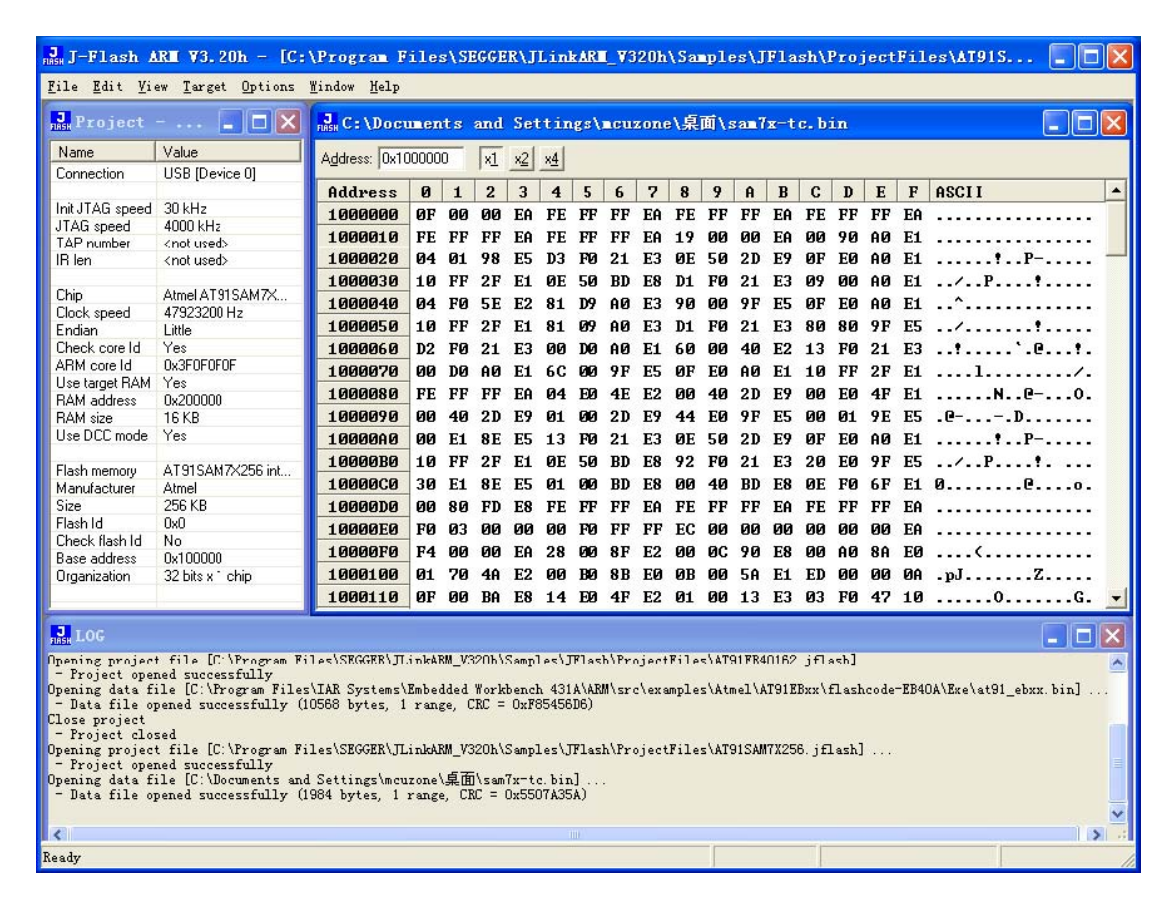
另外，IAR 下使用 JLINK 的时候，注意不要再使用 IAR 自带的 FLASHLOADER 软件进行 FLASH 下载：



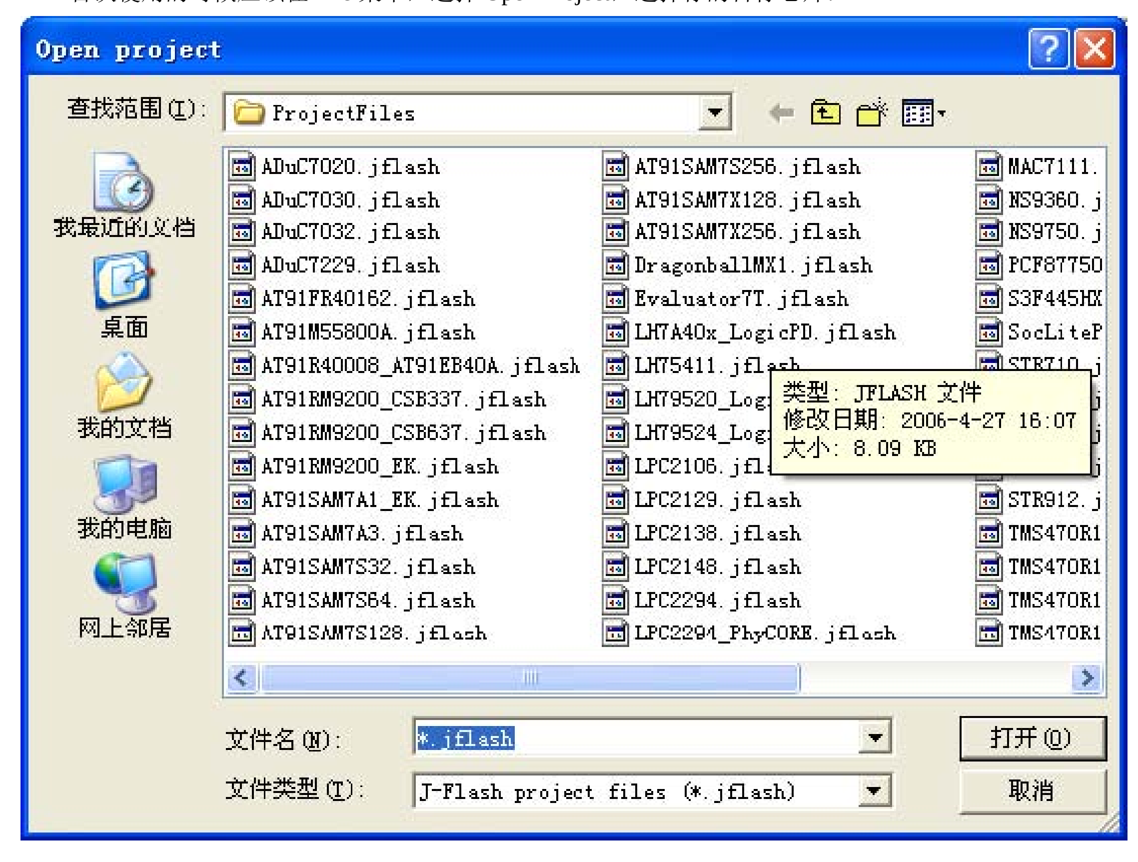
请将“Use flash loader”前的勾去掉，使用 JLINK 的 FLASH 编程算法和使用 IAR 的 FLASHLOADER，速度可能差好几倍！

4. J-FLASH ARM使用设置

安装完 JLINK 的驱动后会出现两个快捷图标，其中一个是 J-FLASH ARM，这个应用程序是用来单独编程 FLASH 的(需要 J- FLASH ARM License 支持)：



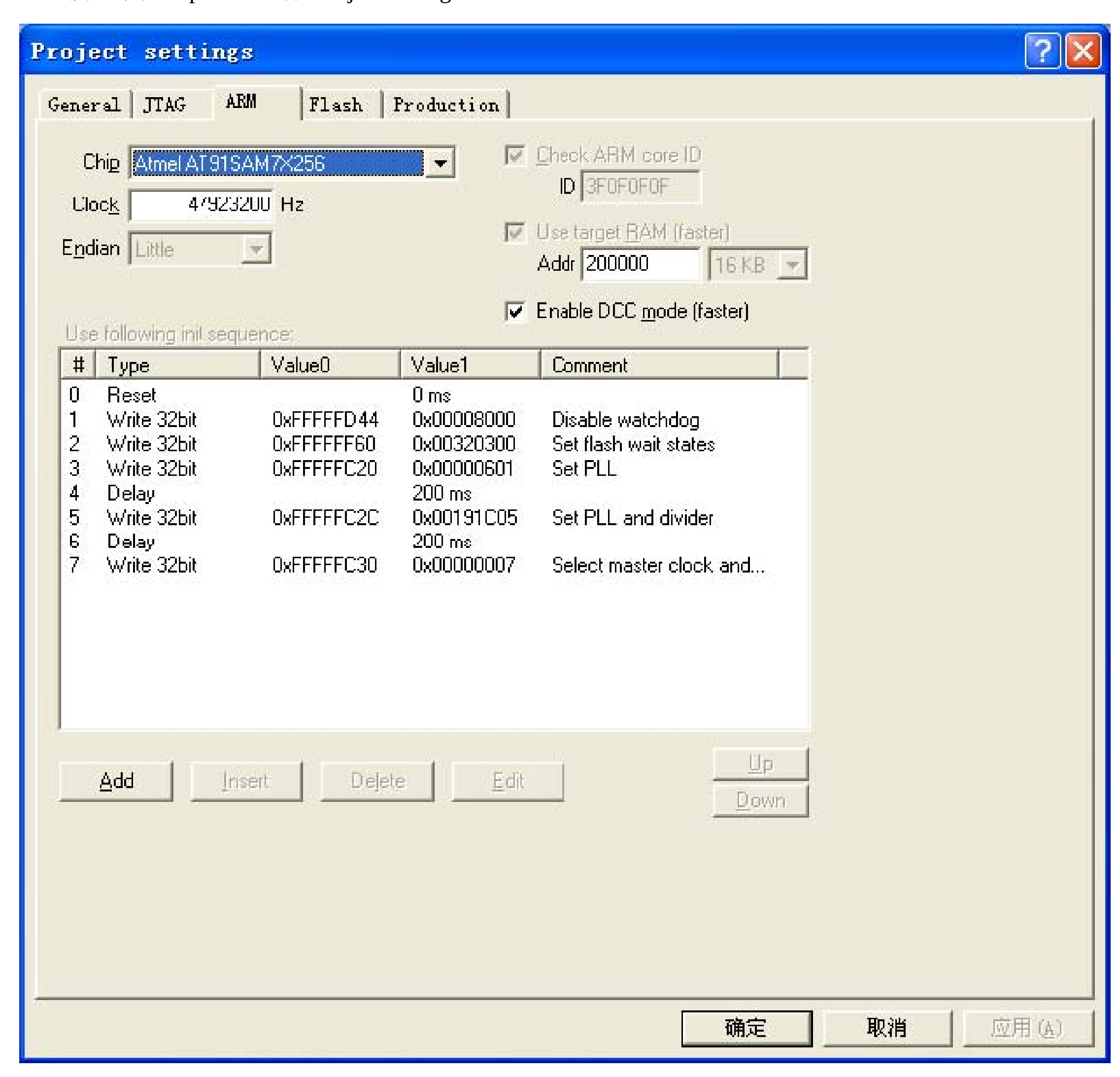
首次使用的时候应该在 File 菜单，选择 Open Project，选择你的目标芯片：



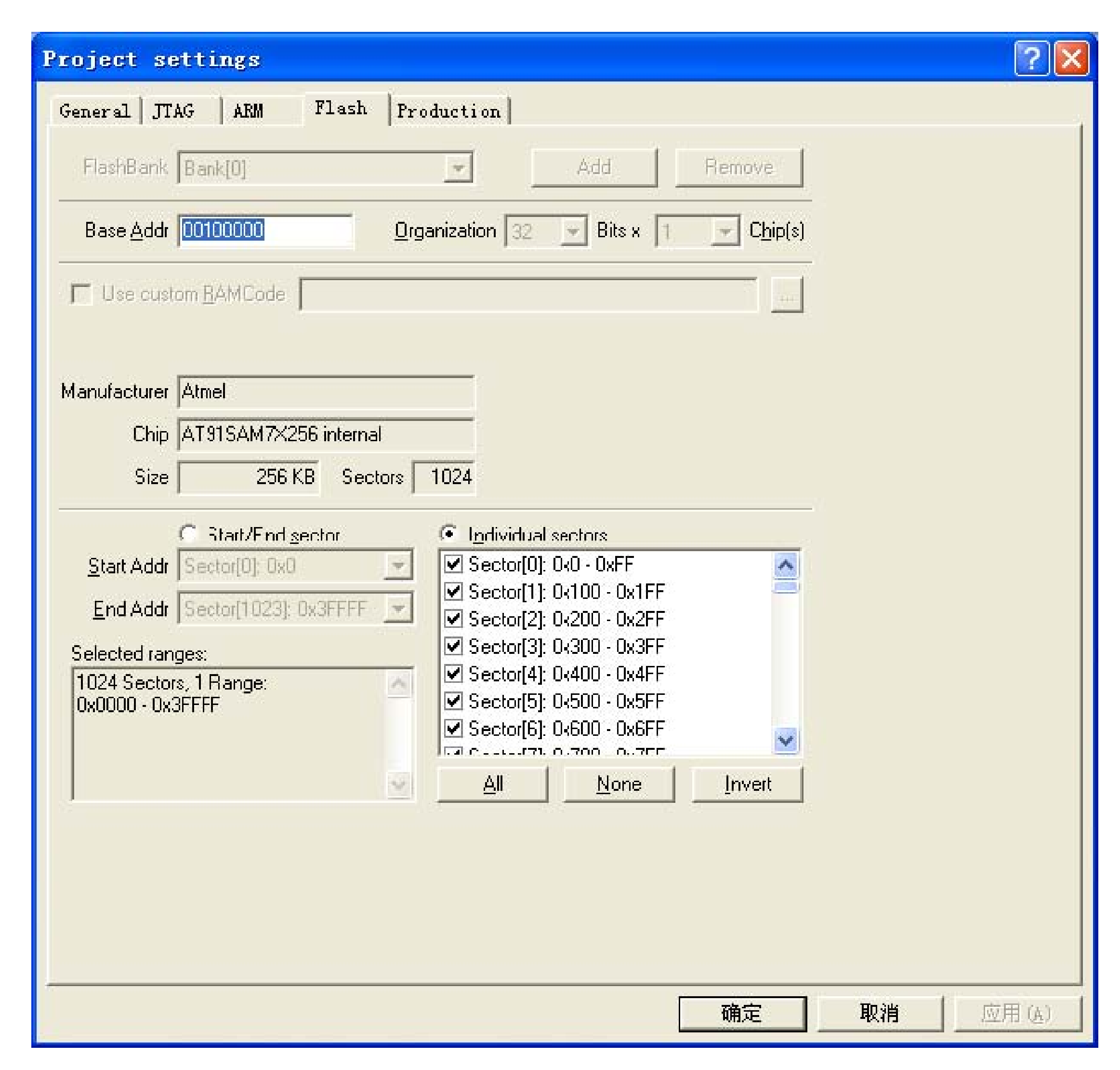
然后通过“File”菜单下的“Open…”来打开需要烧写的文件，可以是.bin 格式，也可以是.hex 格式，甚

至可以是.mot 格式。注意起始地址。

接下来在“Options”选择“Project settings”：

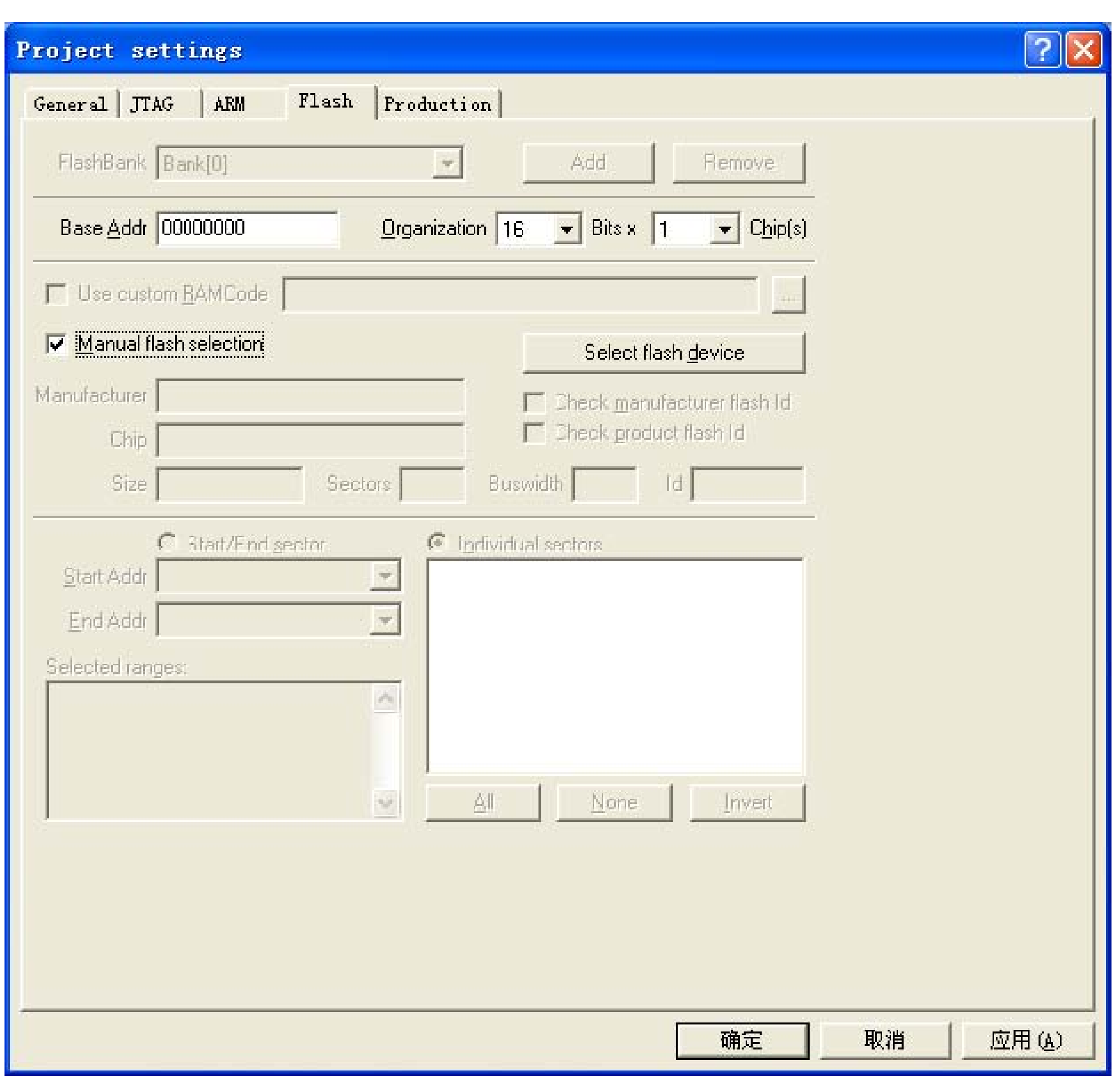


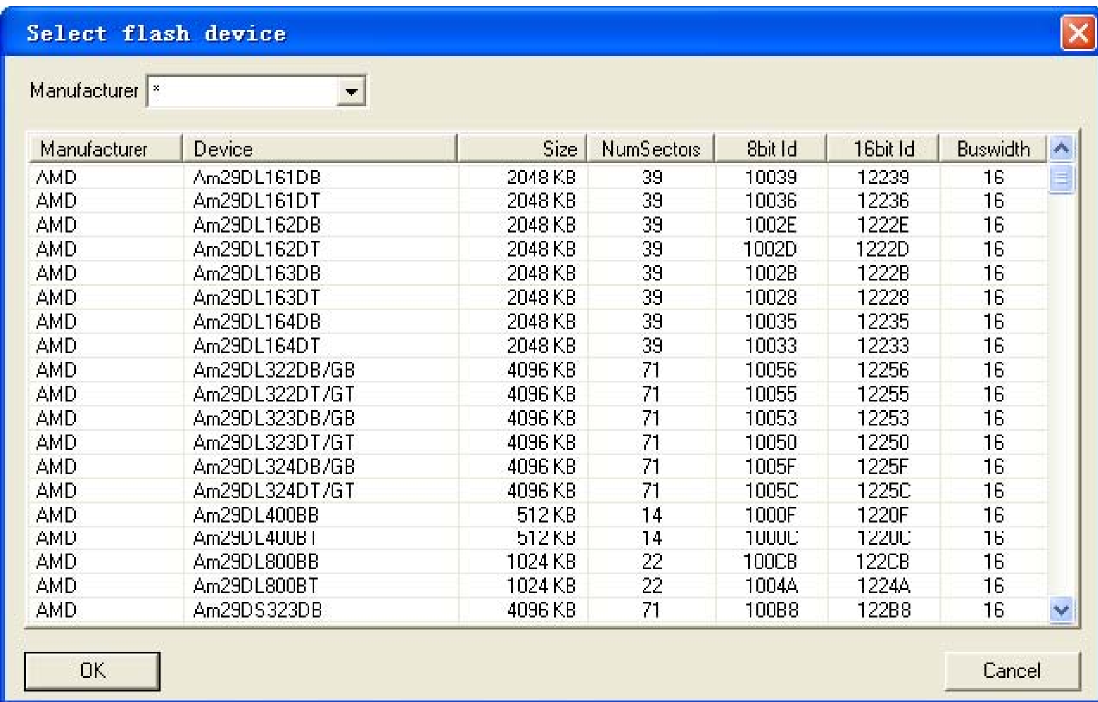
在 ARM 选项卡可以选择目标芯片，如果不是具备片内 FLASH 的芯片的话请选择“Generic ARM7/ARM9”。



FLASH 选项卡，如果之前是“Opon project”这里就不需要设置，默认即可，如果是自己新建的 project，则需要小心设置。

如果前面的 ARM 选项卡里选择的是“Generic ARM7/ARM9”，则可以在 FLASH 选项卡里面选择 FLASH 型号：





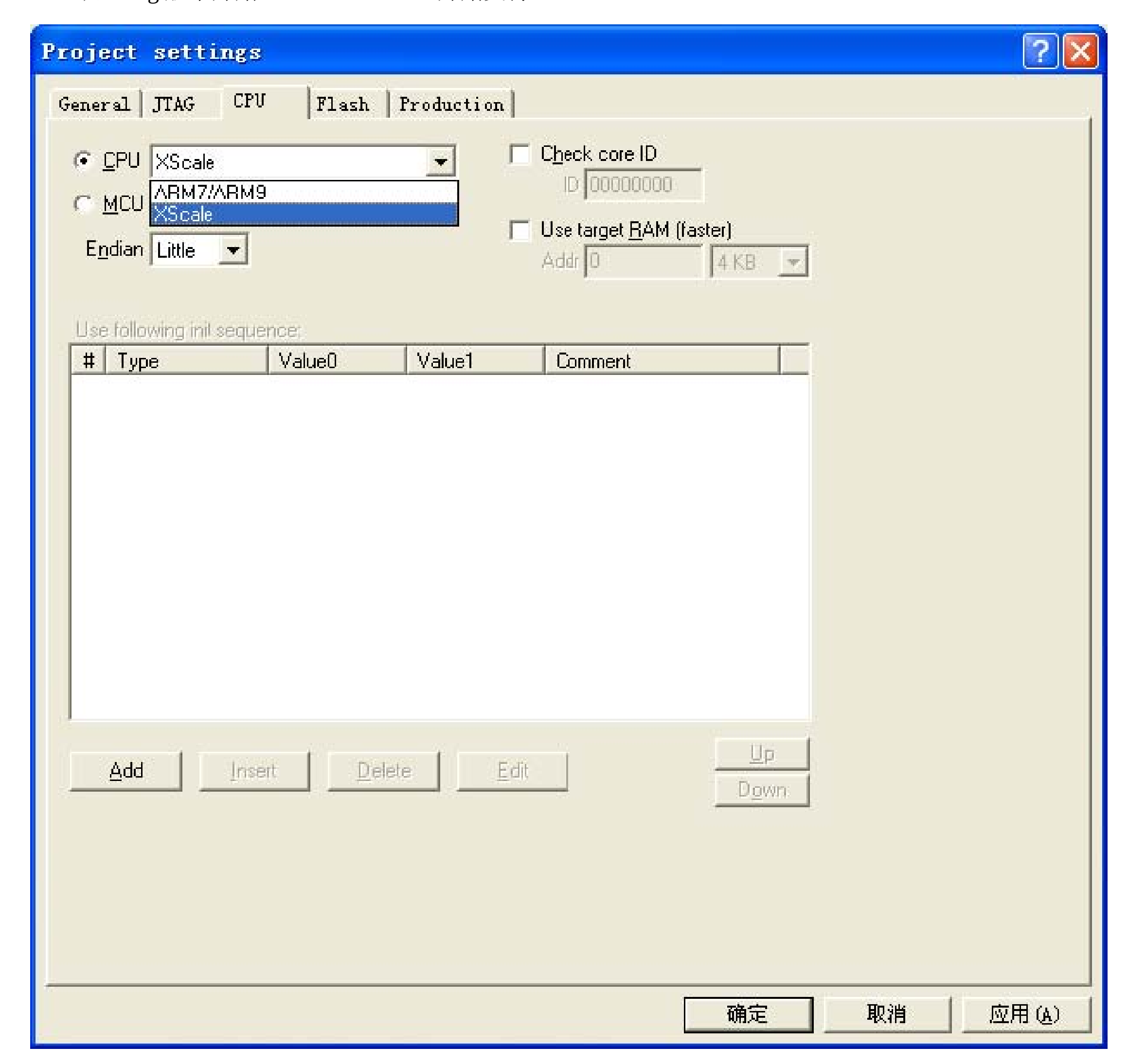
Manufacturer	Device	Size	NumSectors	8bit Id	16bit Id	Buswidth
AMD	Am29DL161DB	2048 KB	39	10039	12239	16
AMD	Am29DL161DT	2048 KB	39	10036	12236	16
AMD	Am29DL162DB	2048 KB	39	1002E	1222E	16
AMD	Am29DL162DT	2048 KB	39	1002D	1222D	16
AMD	Am29DL163DB	2048 KB	39	1002B	1222B	16
AMD	Am29DL163DT	2048 KB	39	10028	12228	16
AMD	Am29DL164DB	2048 KB	39	10035	12235	16
AMD	Am29DL164DT	2048 KB	39	10033	12233	16
AMD	Am29DL322DB/GB	4096 KB	71	10056	12256	16
AMD	Am29DL322DT/GT	4096 KB	71	10055	12255	16
AMD	Am29DL323DB/GB	4096 KB	71	10053	12253	16
AMD	Am29DL323DT/GT	4096 KB	71	10050	12250	16
AMD	Am29DL324DB/GB	4096 KB	71	1005F	1225F	16
AMD	Am29DL324DT/GT	4096 KB	71	1005C	1225C	16
AMD	Am29DL400BB	512 KB	14	1000F	1220F	16
AMD	Am29DL400BT	512 KB	14	1000C	1220C	16
AMD	Am29DL800BB	1024 KB	22	100CB	122CB	16
AMD	Am29DL800BT	1024 KB	22	1004A	1224A	16
AMD	Am29DS323DB	4096 KB	71	100B8	122B8	16

支持非常多的 FLASH 器件，只要是大厂的 FLASH，基本都可以找到！而且会不断升级以支持最新

器件。

设置好之后，就可以到 Target 里面进行操作，一般步骤是先“Connect”，然后“Erase Chip”，然后“Program”，可以自己慢慢体会。大部分芯片还可以加密，主要的操作都在 Target 菜单下完成。

从 3.30g 版本开始，J-FLASH ARM 开始支持 XSCALE：



5. JLINK　F.A.Q

1Q：国内代理商卖的 JLINK 价格是 1900，贵站兼容 JLINK 的调试器的价格也在 1000 多，价格优势似乎不明显？

1A：注意，JLINK 分很多版本，国内代理商销售的 1900 的 JLINK 全是 IAR 版本的，即只能在 IAR 下使用，而且功能有所限制，速度也有限制！本站可以按照客户需求进行定制，最全版本的 JLINK 的价格不超过 2000，但是同样功能的原装 JLINK 的价格是 1000 欧元！而且目前国内仅 MCU123 一家可以提供全功能版本的 JLINK，市面上在销售的其他 JLINK 全部都是 IAR 版本的，即 SEGGER 为 IAR 做的 OEM 产品！同样，ATMEL 的 SAM-ICE 也是 SEGGER 做的 OEM 产品，但是限制更多。本站也可以按照客户需求进行定制，IAR 版本的 JLINK 的价格在 1000 以内。

2Q：JLINK 和其他 JTAG 调试工具相比有什么优势？

2A：全功能版本的 JLINK 具有如下主要特点：

1）支持 ADS,KEIL,IAR,WINARM,RV 等几乎所有开发环境（RDI License 支持）；

2）支持 FLASH 软件断点，突破一般 ARM 仿真器 2 个 FLASH 断点的限制，可以设置无穷个 FLASH 断点，极大的提高调试效率（Flash BP License 支持）；

3）支持 FLASH 编程，可以在各个开发环境下轻松编程 FLASH（Flash DL License 支持）；

4)具备单独烧写 FLASH 的独立软件，提高生产效率(J-FLASH ARM License 和 J-FLASH ARM 软件支持)；

5）超快速度，编程速度和调试速度在目前已知调试工具里面最快（达到 600K，请参考 JLINK 用户手册）；

6）支持几乎所有 ARM7,ARM9，暂时不支持 XSCALE（支持器件列表请参考 JLINK 用户手册），从 3.30g 版本开始 J-FLASH ARM 软件已经可以支持 XSCALE 系统的 FLASH 编程；

目前，ULINK（SMARTDEBUGGER）只能在 KEIL 下使用；MULTI-ICE（本站提供并口，USB 两个版本）可以在 ADS、IAR 下使用，在 IAR 下使用的时候可以利用 IAR 的 FLASHLOADER 进行 FLASH 编程，但是在 ADS 下使用的时候缺少编程插件；EASYJTAG 只能在 ADS 下使用；WIGGLER 可以在各个开发环境下使用，但是目前只能在 IAR 下用 MACRAIGOR 的驱动，才能编程 FLASH，而且速度很慢；而 JLINK 可以在各种开发环境下调试、下载程序！

3Q：JLINK 提供升级以支持新器件么？

3A：可以到www.segger.com网站下载JLINK（JLINK）安装程序（驱动），segger升级较快，请密切关注。如果需要更改JLINK的授权，比如将IAR版本升级到全功能版本，请直接发回给我们进行升级，最终补版本差价即可。

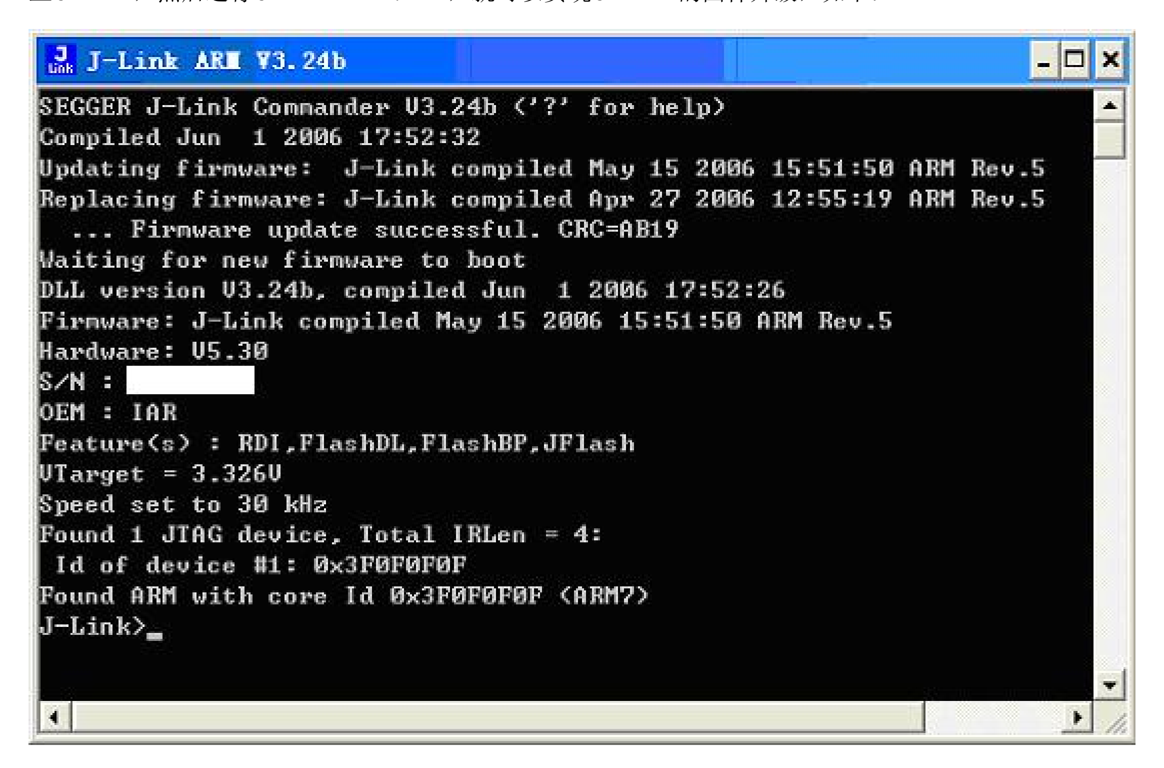
4Q：为什么我购买的 JLINK 在 KEIL 以下不能使用，出现以下错误：



4A：J-LINK 提示没有相应的 license，亦即没有授权，亦即您购买的 J-LINK 并不附带 RDI License，是 IAR 版本的 J-LINK，如果需要 RDI 接口的 License 可以发回给我们进行升级，升级按照功能收费。

5Q：J-Link 的驱动程序和应用程序是否可以免费升级？

5A：Segger 网站升级较快，建议大家经常关注一下，J-Link 的驱动程序和应用程序是免费升级的，只要你购买了某个功能的 License，该部分功能就可以永远免费升级，当下载了新版本的 J-LINK 程序后，只要插上 J-LINK，然后运行 J-LINK ARM.EXE，就可以实现 J-LINK 的固件升级，如下：



请注意看图中的第 3—5 行的信息，软件提示升级成功。

6Q：我在 KEIL 下面调试 LPC2142，为了达到最快的速度，我在 Configure 里面将 JTAG 速度从 AUTO 修改到 12M，但是系统提示：

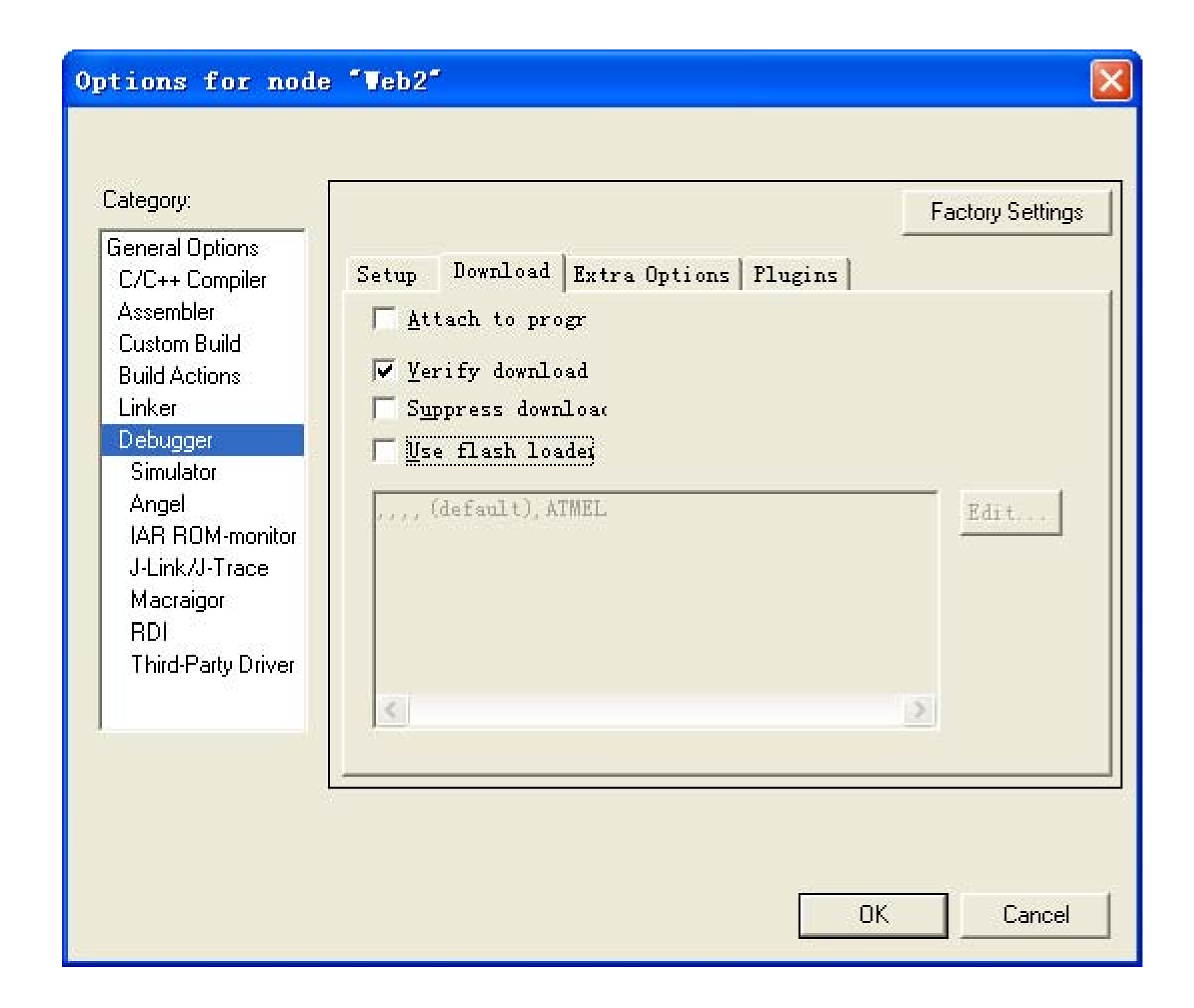


请问这个是什么问题？如何解决？另外，用 AUTO 的话就没有问题。

6A：这个是由 LPC2000 的内核特殊性所决定的。LPC2000 的内核是 ARM7TDMI-S，是可综合版本的 ARM7TDMI，即 PHLIPS 有权限来对 ARM7TDMI 进行部分改动，主要是调试接口的改动，LPC2000 采用的 JTAG 接口包含了一个 RTCK 引脚，这个引脚是用来同步 JTAG 调试时钟用的，当 TCK 发送一个时钟，该时钟经过一定延迟后就由 RTCK 返回，如果接收不到返回的时钟，系统就会提示找不到目标芯片，即调试失败。经过测试，LPC2000 系列 ARM7TDMI-S 最高只能稳定工作在 4800KHz 频率下，再高就会出现以上错误提示。由于 ULINK 使用的最高 JTAG 只能达到 1M，所以在使用 ULINK 的时候根本就不会出现这个问题。从另一个侧面讲，亦即调试 LPC2000 的时候，JLINK 的速度最高可以是 ULINK 的 4.8 倍。

7Q：我使用 IAR 开发环境，为什么用 JLINK 的 FLASH 下载速度和用 MULTI-ICE 的下载速度差不多？

7A：使用 IAR 的时候请注意不要使用 IAR 自带的 FLASHLOADER 进行 FLASH 下载，而应该使用 JLINK 的 FLASH 编程算法，关键一点是将“USE FLASH LOADER”前的勾去掉，如下图：



8A:一般来说 LPC236X 的芯片在未烧录程序时，JTAG 时钟只能在 1M 以下，最好是 500KHZ 左右.

8Q:我用 LPC236X 芯片，为什么 JLINK 老提示出错.

如在 CONFIG 中设置



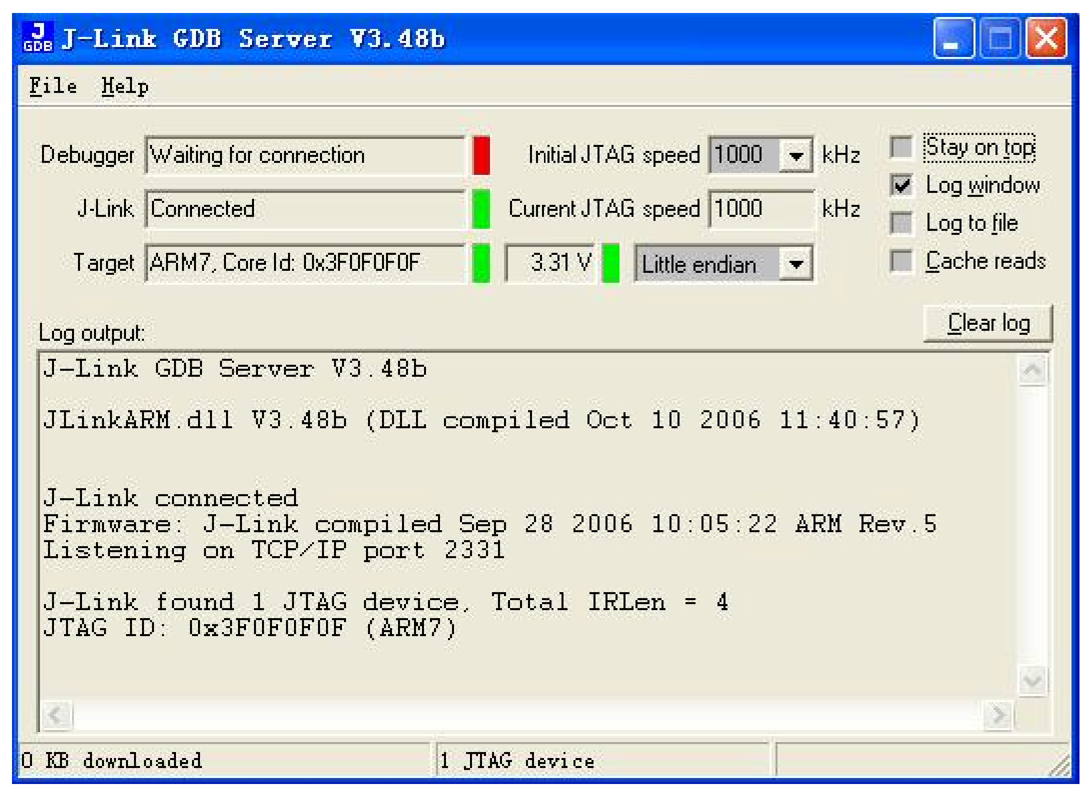
6. 附录一 使用jlink的GDBserver

GDB 作为开源的调试器，其使用比较广泛，是使用 gcc 的标配调试器。

在 segger 官方推出 GDBserver 之前，网上也有个人提供的 jlinkgdbserver，但是效果不是很好。需要注意的是，后者在一般的 jlink 上即可使用，而 segger 官方的还需要一个 GDBfull license 的授权，需要额外的费用。

请先到http://www.segger.com/download_gdb.html下载包含有GDBserver的软件，然后安装。

安装完成后，请连接好 jlink 与目标板，在 pc 端运行 jlink GDBserver，正确的显示如图：



可以看到目标器件的类型，ID， 目标板电压等。

说明此时GDBserver已经与目标器件建立了联系，等待GDB从端口2331来连接。

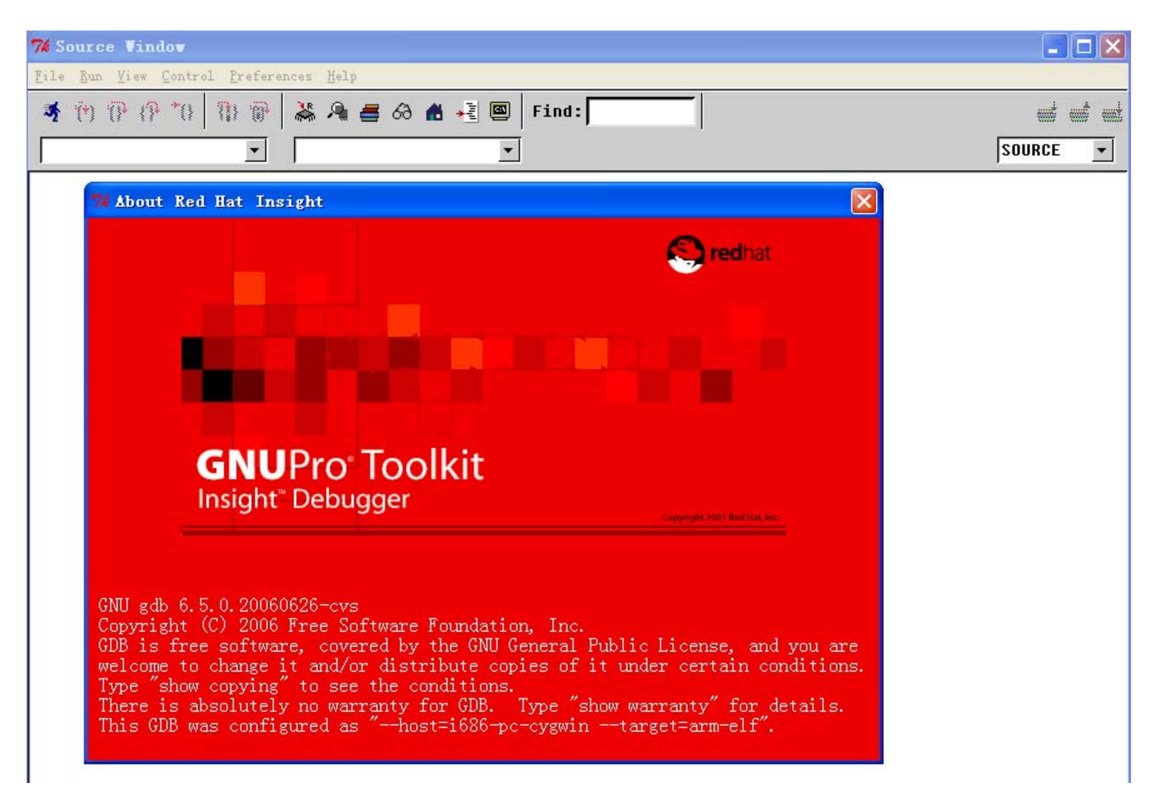
为了使用 GCCARM 来编译软件，还需要安装 GNUARM 或者 WinARM。

调试

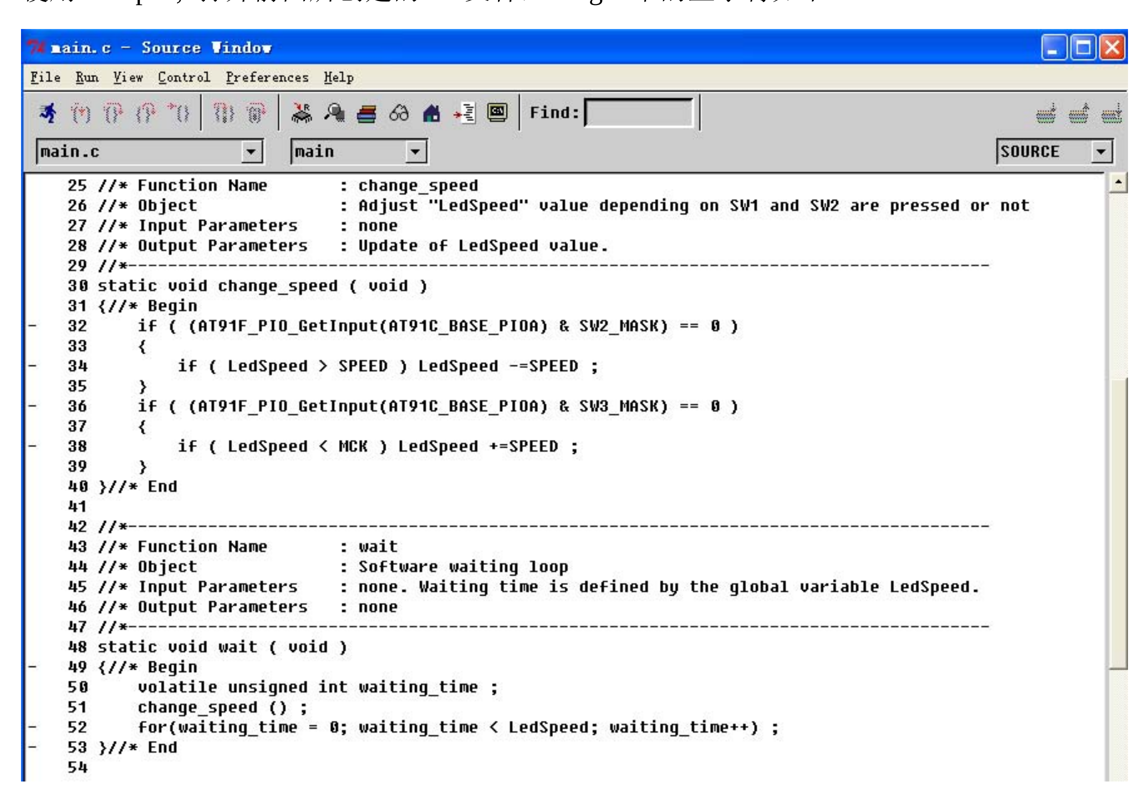
使用 GCCARM 编译应用，最终会生成一个 elf 文件，注意在编译的时候要打开调试信息的选项，比如使用参数－gdwarf－2。

运行 arm-elf-insight，这是个图形化的 ARM GDB，如下图：

然后点击 runrun，在弹出的 target select 中按照如下设置

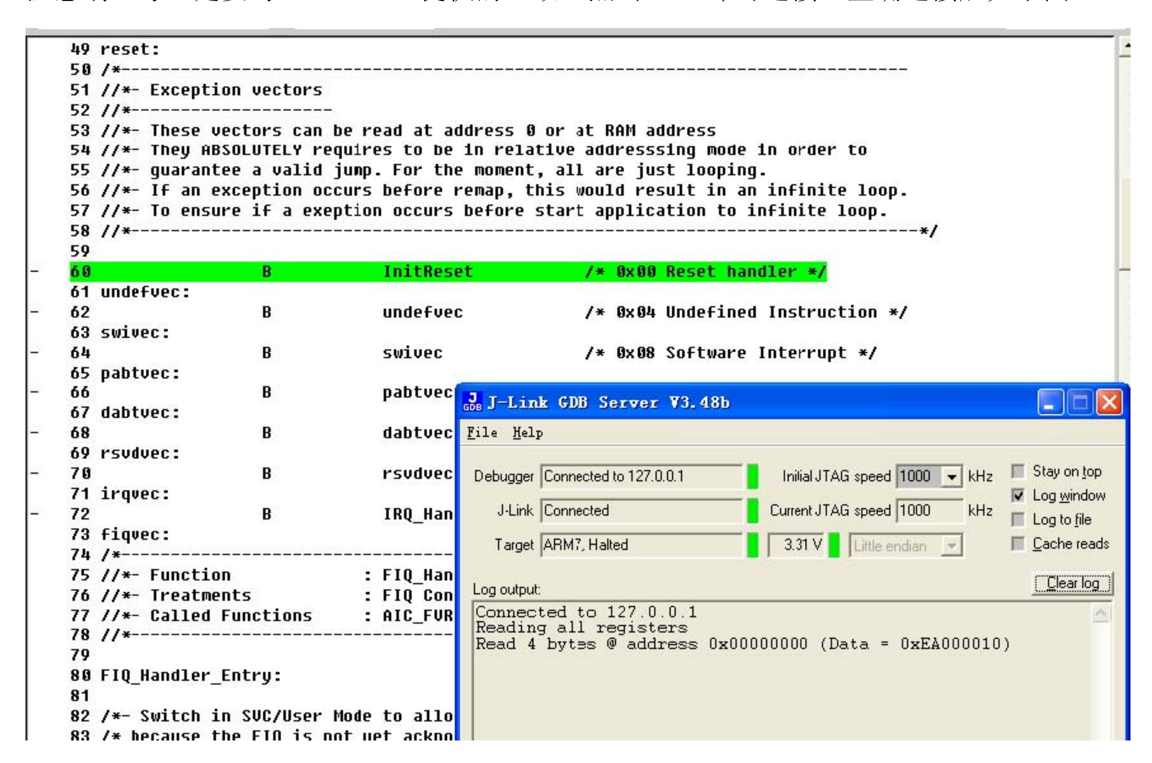


使用 fileopen，打开前面所创建的 elf 文件，insight 中的显示将如下



然后点击 runrun，在弹出的 target select 中按照如下设置

注意端口号一定要与 GDBserver 提供的一致。点击 ok，即可连接。正确连接后如下图：



GDBserver 中会显示已连接，同时 insight 中指令也会停在起始位置。此时就可以开始调试了，比如按 s 单步进入。

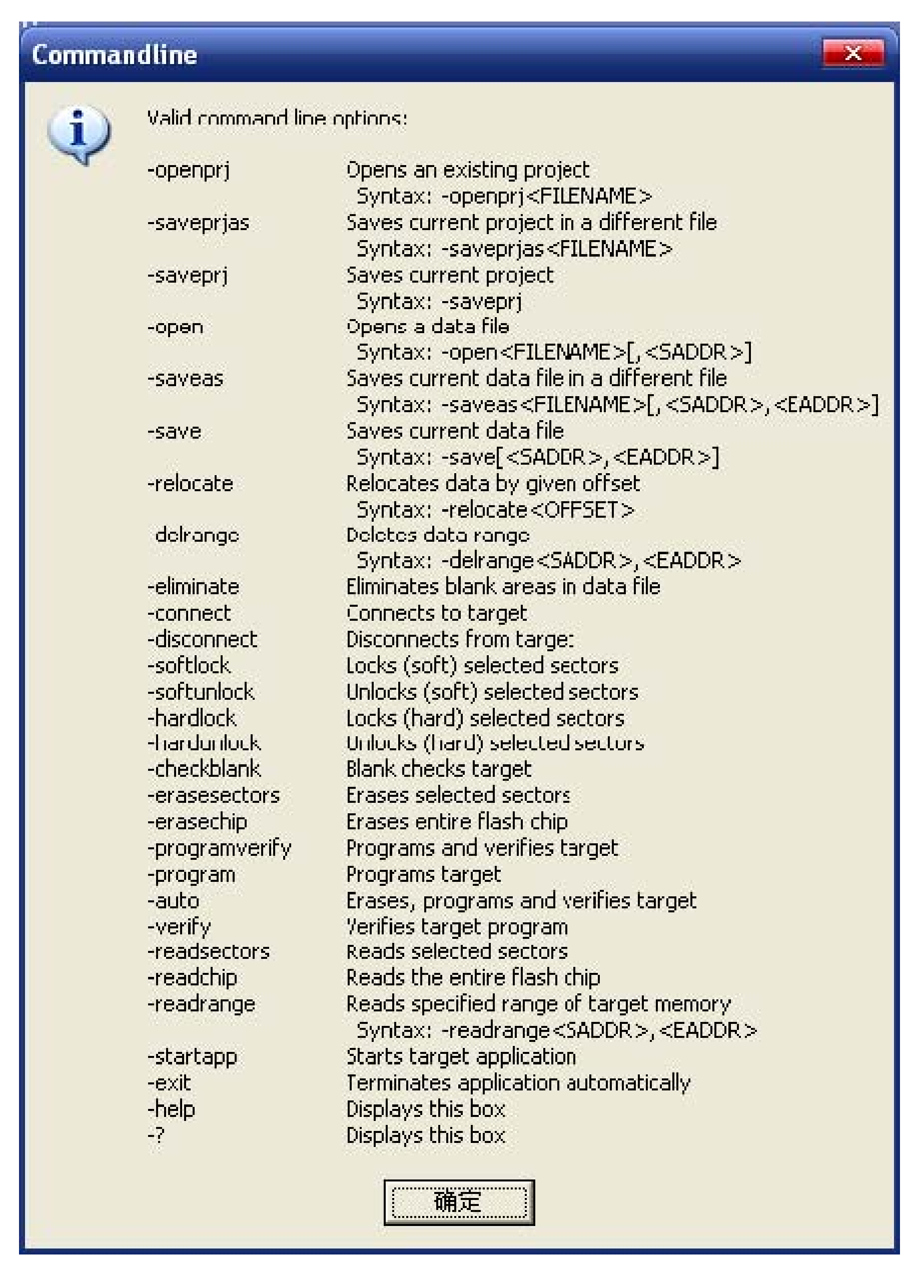
由此可见，新加入的 GDBfull license 对 GDB 的支持更好，使用其来调试也较方便。

7. 附录二、J-Flash ARM命令行使用说明

为了方便扩展使用，J-Flash ARM 还提供了命令行方式。

需要注意的是，默认安装目录是 program files 文件夹下，而这个路径存在一个空格（即 program 和 files 中间的空格），而这在命令行方式下是不允许的，所以，如果使用命令行，需要更改安装路径，或者把工程文件和目标文件放到别的目录下。

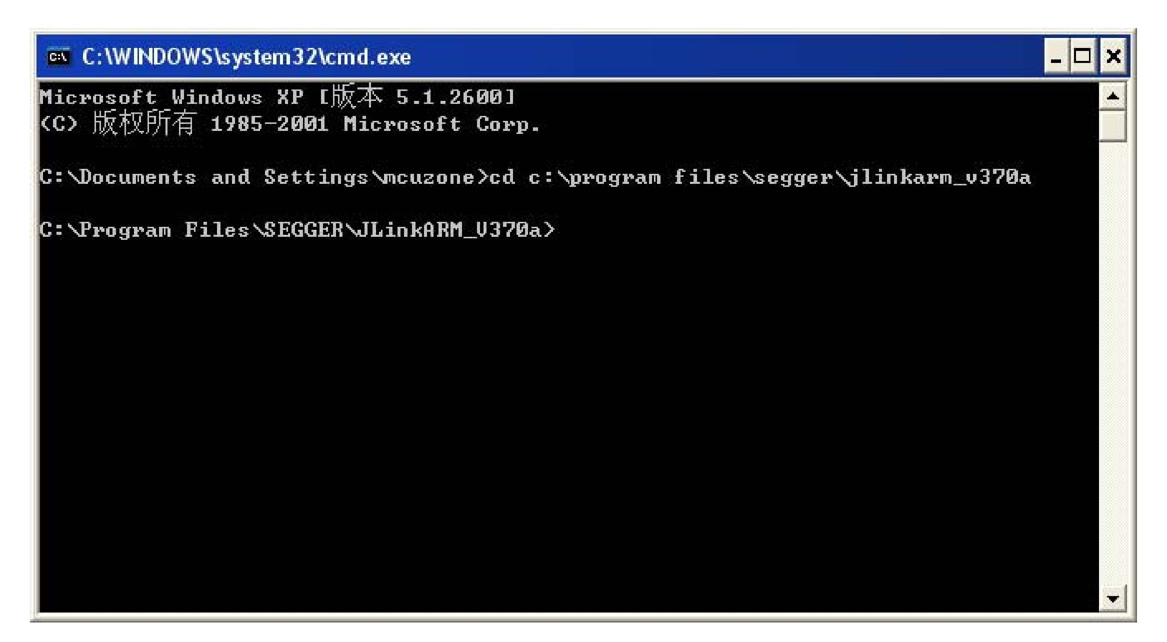
J-FLASH ARM 主要有以下命令：



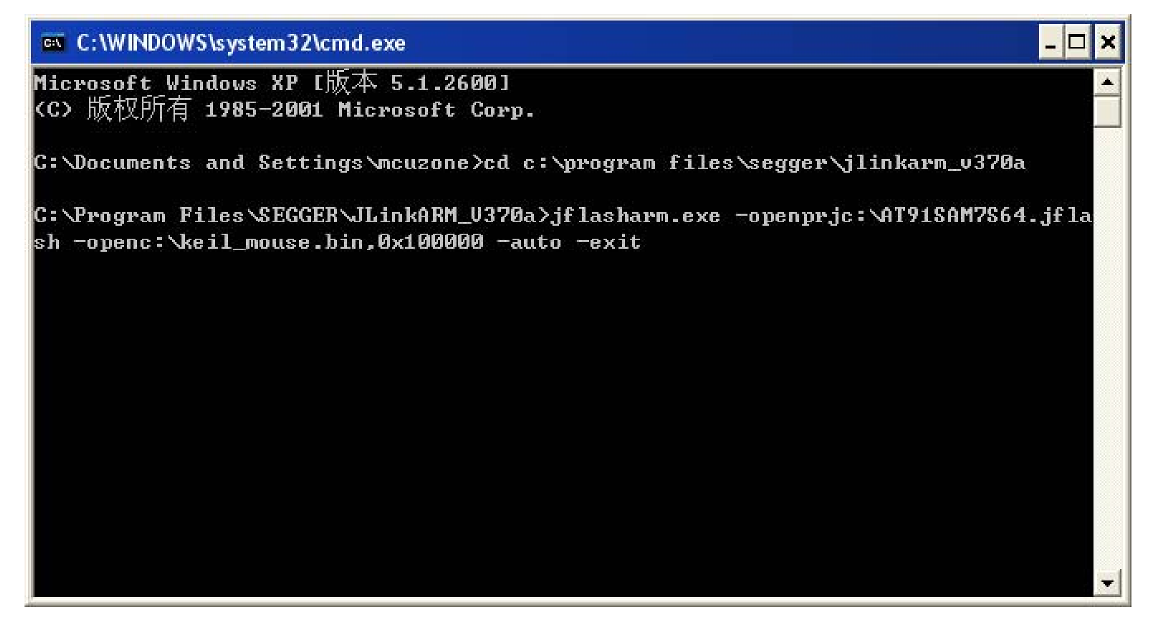
下面我们以 AT91SAM7S64 为目标芯片，来进行命令行演示。

进入命令行状态前，我们先把 AT91SAM7S64.JFLASH 工程文件和 KEIL_MOUSE.BIN 文件放到 C 盘根目录下，方便操作。然后连接好目标板和 J-LINK。

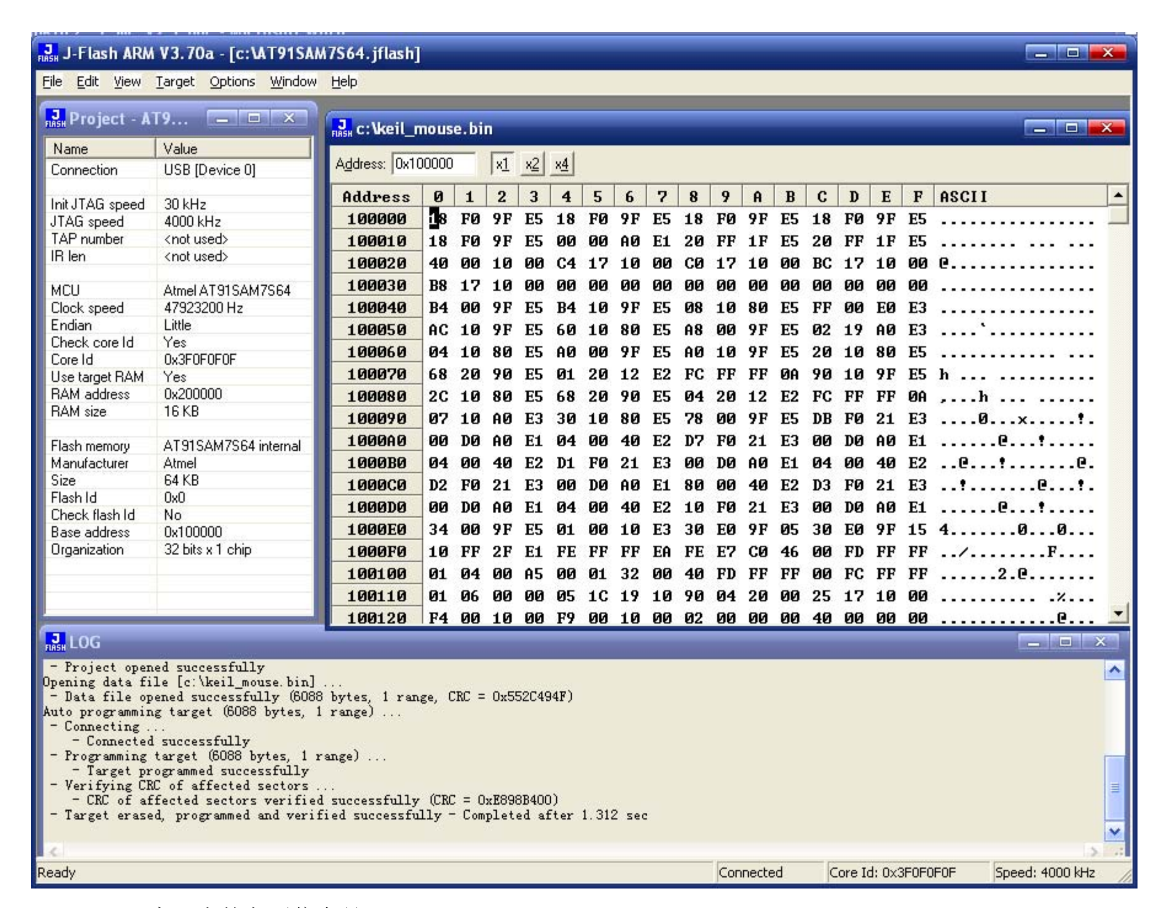
首先进入到安装目录：



然后键入命令，如下图所示：



回车后，J-FLASH ARM 的用户界面会被弹出，然后可以看到 J-FLASH ARM 很快完成操作并退出，如果我们要看整个操作过程的 log 信息，我们可以去掉命令行的-exit 参数，去掉这个参数后，J-FLASH ARM 在完成操作后并不会被关闭，这个时候我们可以通过 log 窗口看到操作信息：



Log 窗口内的主要信息是：

Opening project file [c:\AT91SAM7S64.jflash] ...

- Project opened successfully

Opening data file [c:\keil_mouse.bin] ...

- Data file opened successfully (6088 bytes, 1 range, CRC = 0x552C494F)

Auto programming target (6088 bytes, 1 range) ...

Connecting ...

Connected successfully

Programming target (6088 bytes, 1 range) ...

Target programmed successfully

Verifying CRC of affected sectors ...

CRC of affected sectors verified successfully (CRC = 0xE898B400)

Target erased, programmed and verified successfully - Completed after 1.312 sec

可以看到-openprj 命令就是打开工程文件，即 FLASH 编程算法；-open 是打开数据文件，即需要写入的 bin 或者 hex 文件，需要注意的是-open 参数后面还需要添加烧写地址，即上述命令里面的“,0x100000”，不然会编程失败；-auto 是指自动操作，包含了擦除，编程，校验几个步骤。如果只需要读，擦除，编程等一个单独的操作，J-FLASH ARM 也提供了相应的命令参数，可以自行尝试。在尝试阶段，建议不要加-exit 命令，方便查看 log 窗口的信息，以确认操作是否成功完成。

既然提供了命令行方式，我们就可以使用批处理命令来使得操作更为简单：

新建一个文本文件，然后键入以下内容：

“cd c:\program files\segger\jlinkarm_v370a

jflasharm.exe-openprjc:\AT91SAM7S64.jflash -openc:\keil_mouse.bin,0x100000 -auto ”

然后另存为 bat 文件，如 jflash.bat。

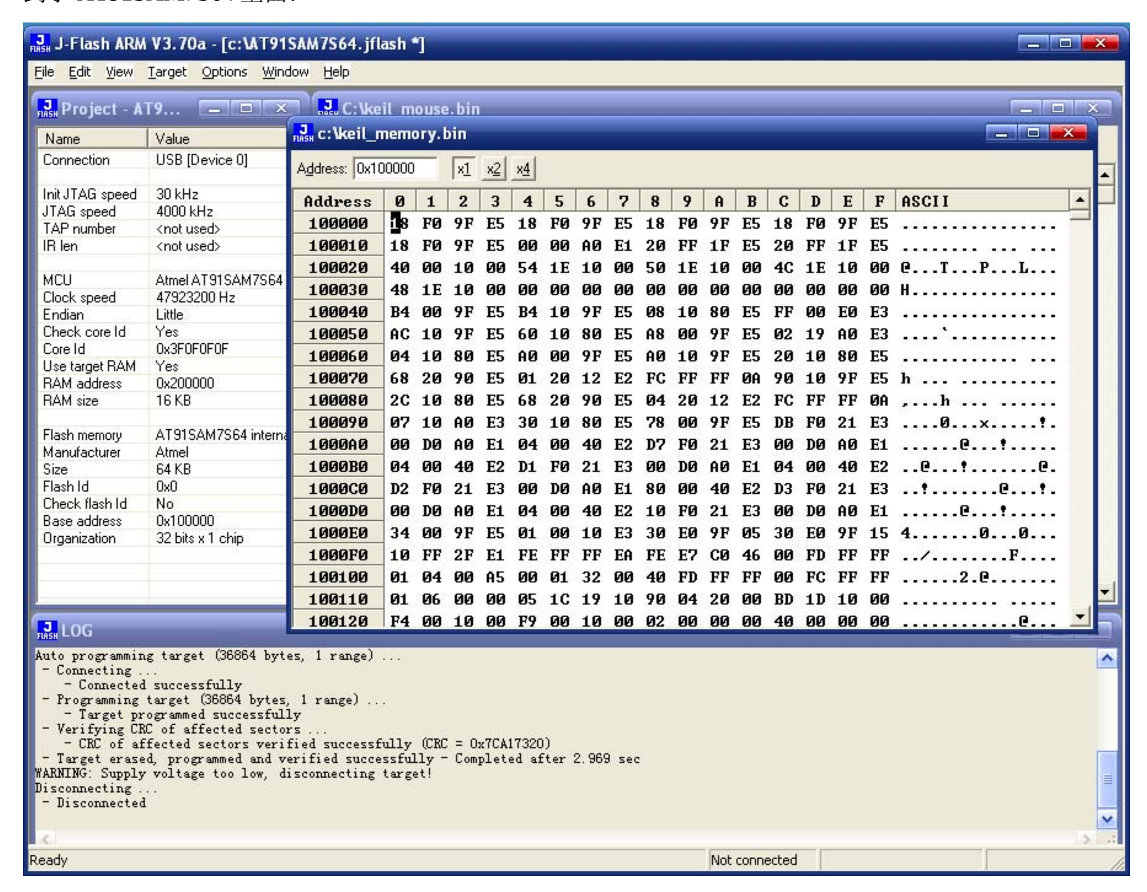
然后运行该 bat 文件，可以获得和前面命令行一样的效果。

更进一步的，我们来挖掘一下 bat 的批处理功能，新建一个 bat 文件，键入以下内容：

“cd c:\program files\segger\jlinkarm_v370a

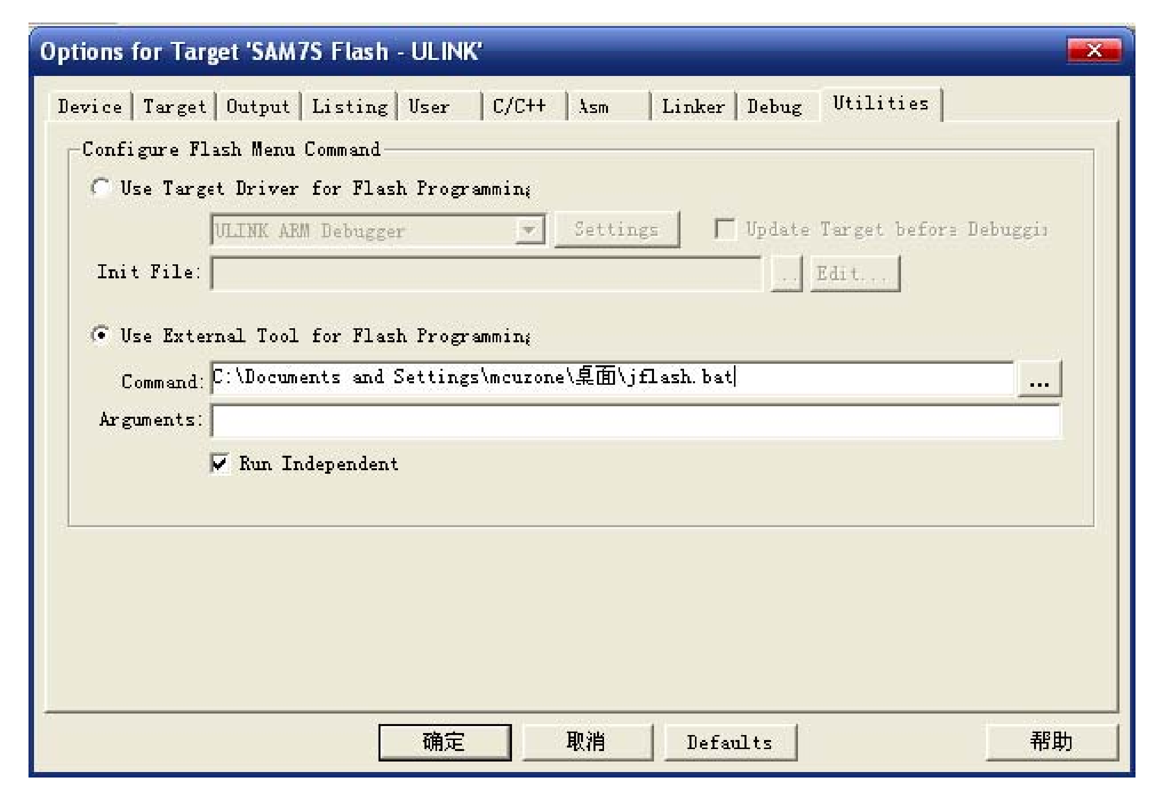
jflasharm.exe -openprjc:\AT91SAM7S64.jflash -openc:\keil_mouse.bin,0x100000 -auto –exit

然后运行 bat 文件，可以看到 J-FLASH 运行了两次，分别把 keil_mouse.bin 和 keil_memory.bin 写入到了 AT91SAM7S64 里面：



利用 bat 的特性我们可以用来完成一些特殊用途。比如对于具备片内 FLASH，同时又开放总线的 ARM 芯片，比如 STR710。我们可以先新建两个工程，分别针对片内 FLASH 和片外 FLASH，然后建立 bat 文件，分别打开两个工程，编程两段 FLASH。这样可以有效提高工作效率。

更进一步，我们可以在 keil 下也加入这个功能，由于 keil 开放了一个外部 FLASH 编程工具接口，使得使用 bat 文件成为可能，打开 Keil 的“Options for Target”选项，选择“Utilities”选项卡，把默认的“Use Target Driver for Flash Programming”换成“Use External Tool for Flash Programming”，然后在“Command”一栏选择之前设定好的 bat 文件，点击确认。



完成以上设置后，点击 Keil 工具栏上的 Load 按钮：



马上会调入 J-FLASH 的编程界面，和直接运行 bat 文件一样效果。

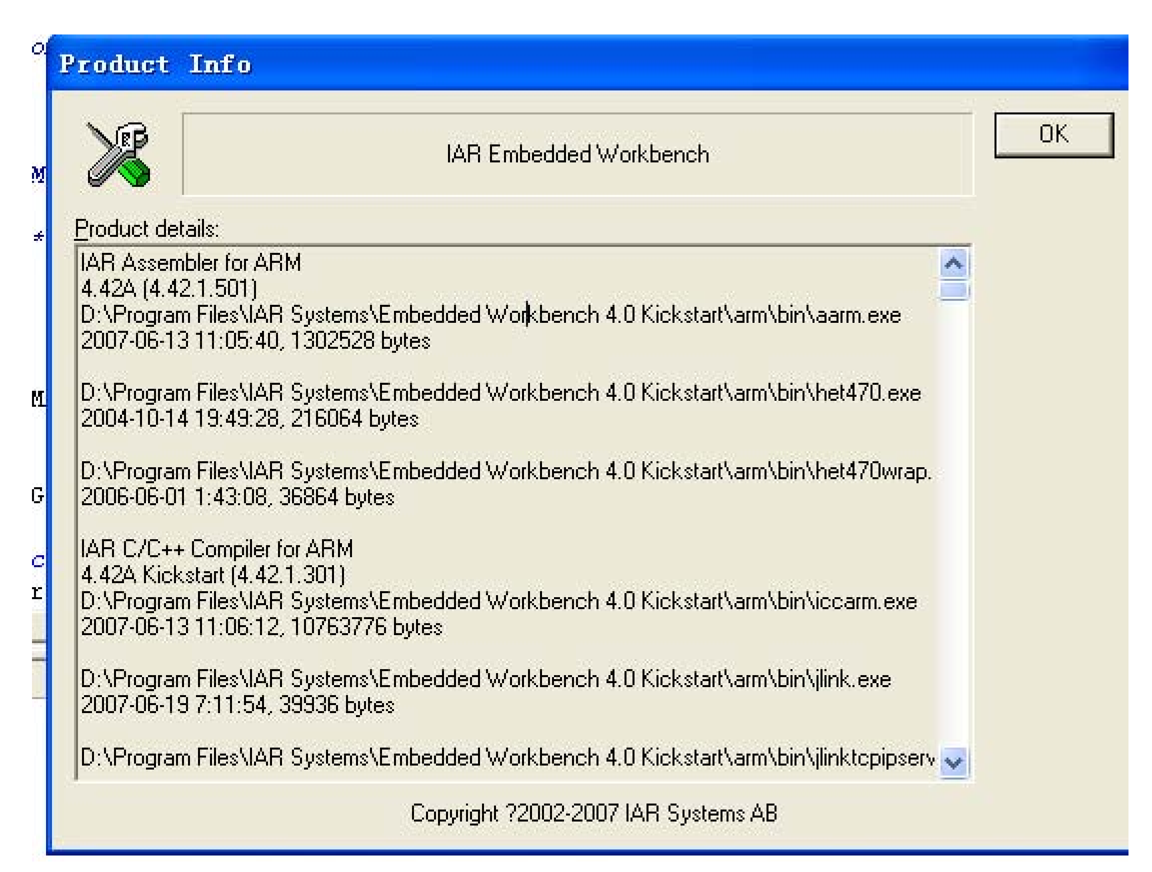
更多的花样和参数可以按照实际需求进行变化和改进，利用批处理的优势可以在调试和批量生产过程中极大的提升效率。

8. 附录三、Jlink 在IAR下调试LM3S系列简易说明

以芯片为 LM3S101 为例。

在 IAR 下有版本的要求：

IAR Embedded Workbench Evaluation for ARM 4.42A 以上。



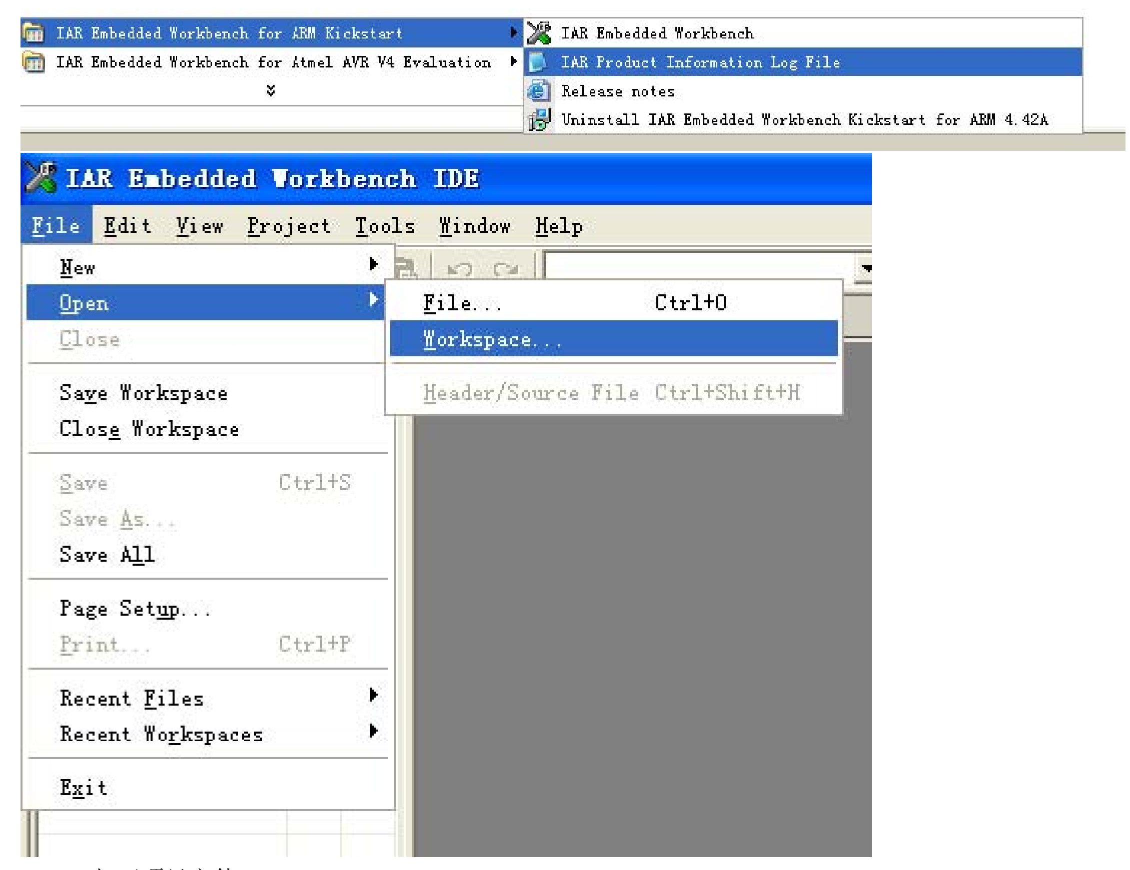
连接好目标板与 JLINK。

然后运行 JLINK commander

```
J-Link Commander
SEGGER J-Link Commander V3.74i ('?' for help)
Compiled Sep 11 2007 14:43:12
DLL version V3.74i, compiled Sep 11 2007 14:43:07
Firmware: J-Link compiled Jun 28 2007 10:45:08 ARM Rev.5
Hardware: V5.40
S/N : 20078817
OEM : IAR
Feature(s) : RDI,FlashDL,FlashBP,JFlash,GDBFull
VTarget = 3.274V
JTAG speed: 30 kHz
Info: Found Cortex-M3, Little endian.
Info: TPIU fitted.
Info:   FPUnit: 6 code (BP) slots and 2 literal slots
Found 1 JTAG device, Total IRLen = 4:
 Id of device #0: 0x2BA00477
Cortex-M3 identified.
J-Link>_
```

可以看到 CM3 内核

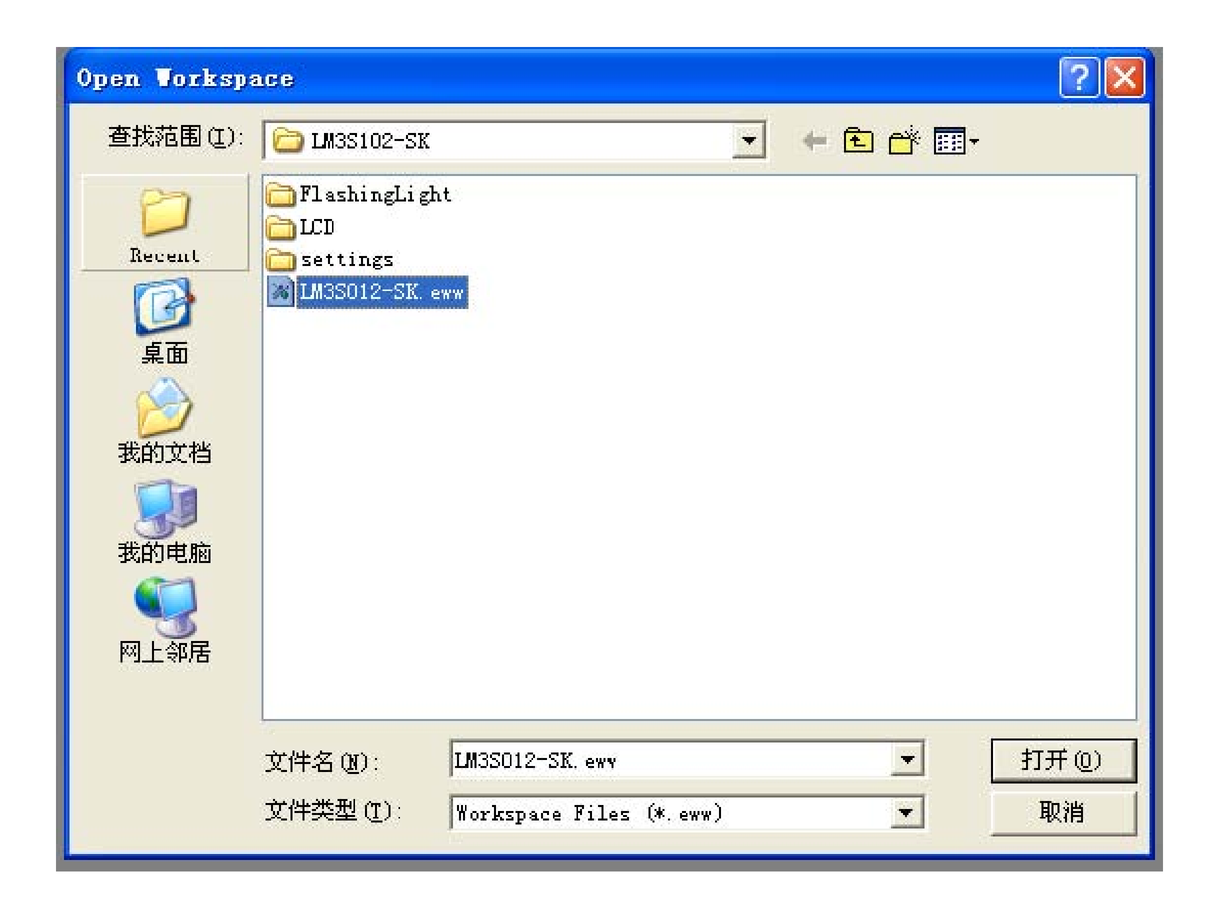
运行 IAR 软件。



打开项目文件。



在 IAR 的安装后的目录中有例程的 D:\Program Files\IAR Systems\Embedded Workbench 4.0 Kickstart\arm\examples\Luminary\LM3S102-SK



打开工程后：

IAR Embedded Workbench IDE

File Edit View Project Tools Window Help

Workspace

LcdDemo - Debug

Files

LM3S012-SK

FlashingLight - ...

LcdDemo - D...

Overview FlashingLight LcdDemo

main.c main.c sysctrl.c

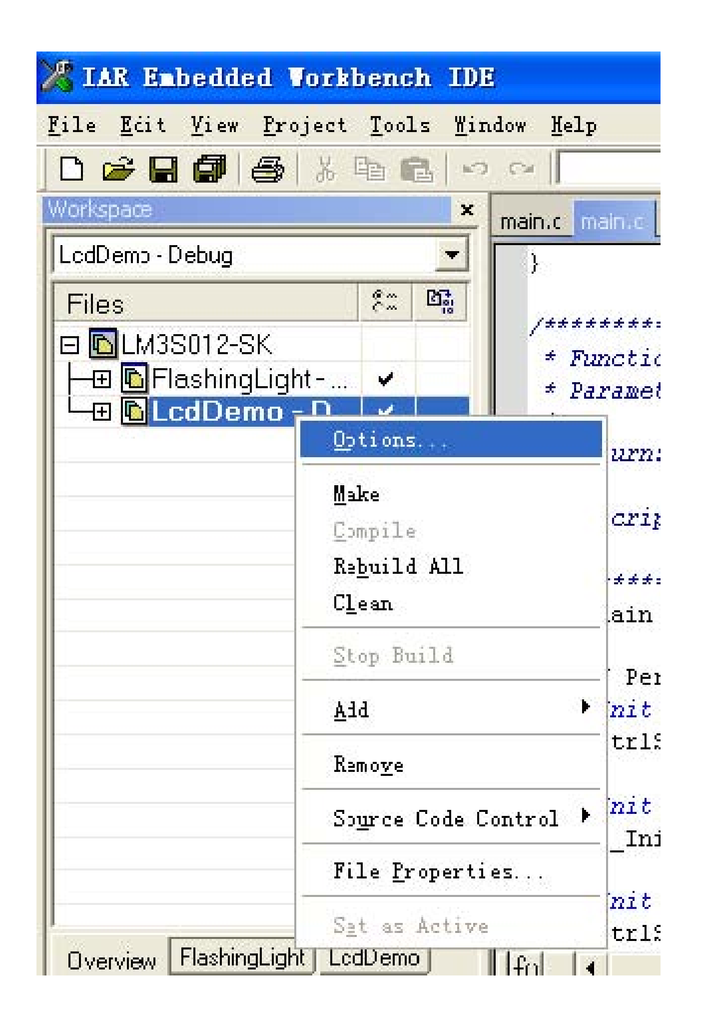
```
}

/*************************************************************************
 * Function Name: main
 * Parameters: none
 *
 * Return: none
 *
 * Description: Main
 *
 *************************************************************************/
void main (void)
{
Int32U Period;
  // Init Clock
  SysCtrlSetClk(MAIN_OSC_TYPE, SYSTEM_DIVIDER, MAIN_OSC_SRC, TRUE, FALSE);

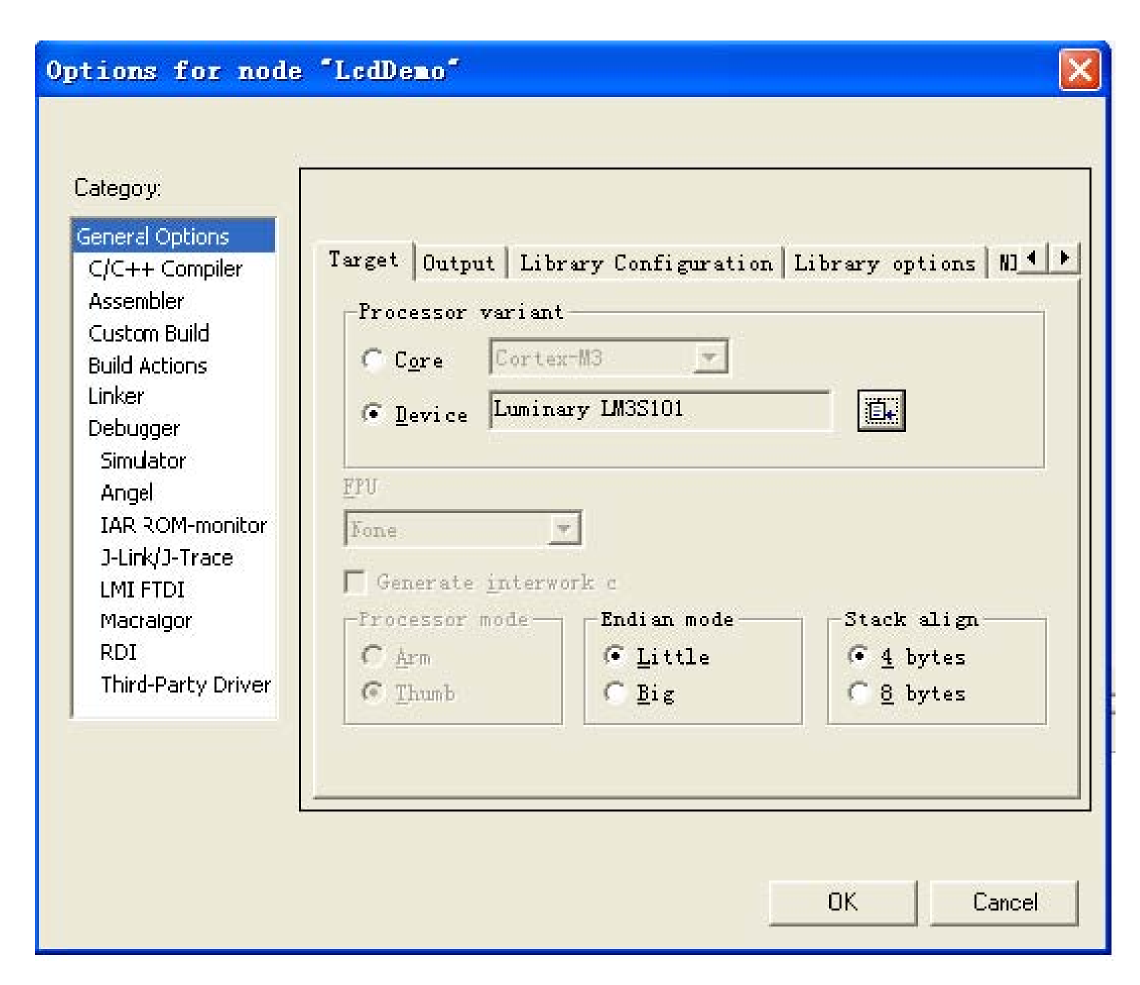
  // Init NVIC
  NVIC_Init(PrioGroup7_1);

  // Init Sys tick timer
  SysCtrlSysTimerInit(TICK_PER_SEC,CORE_CLK);
```

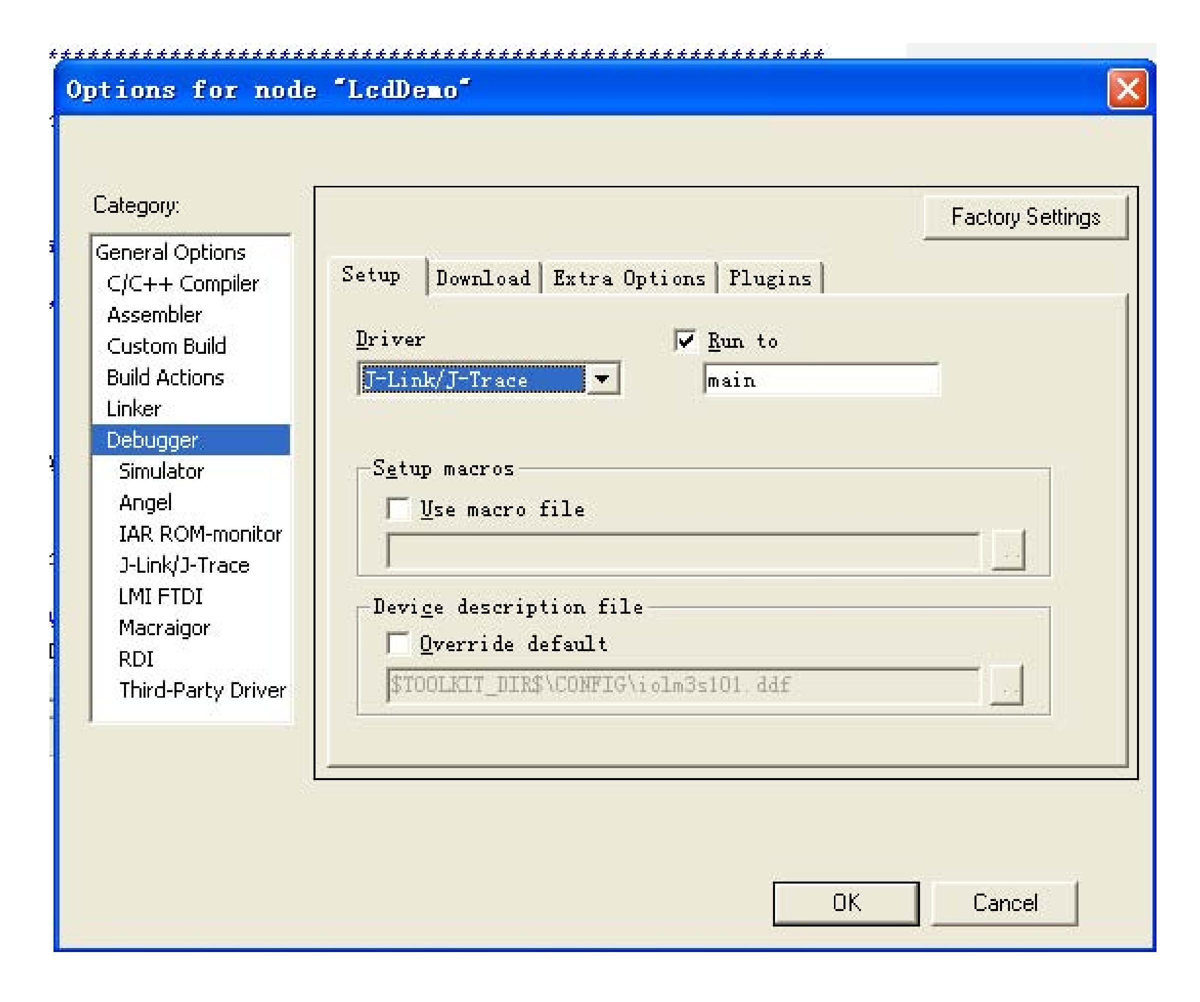
Messages



点击 Lcddemo 右键 option



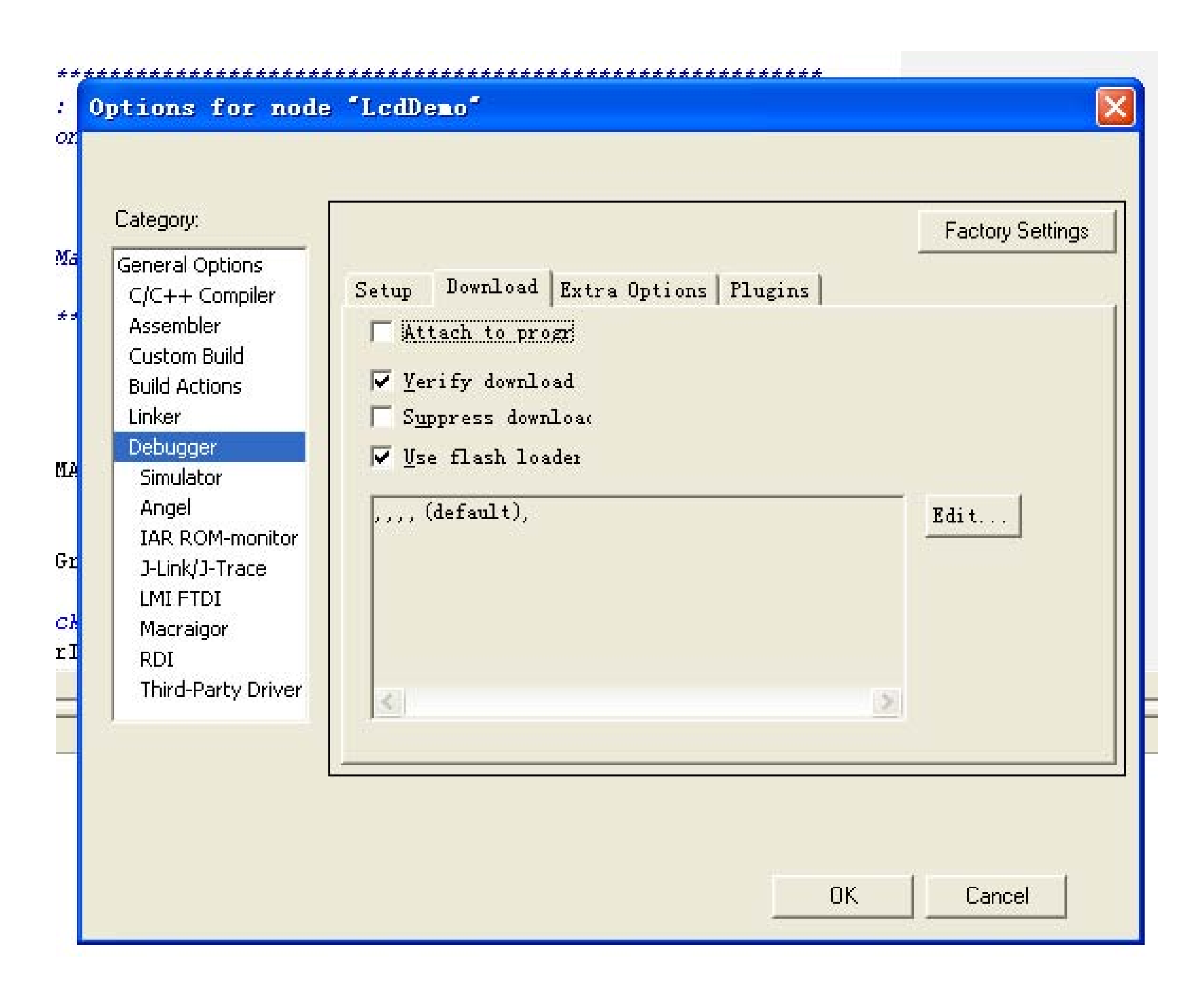
芯片选择：LM3S101

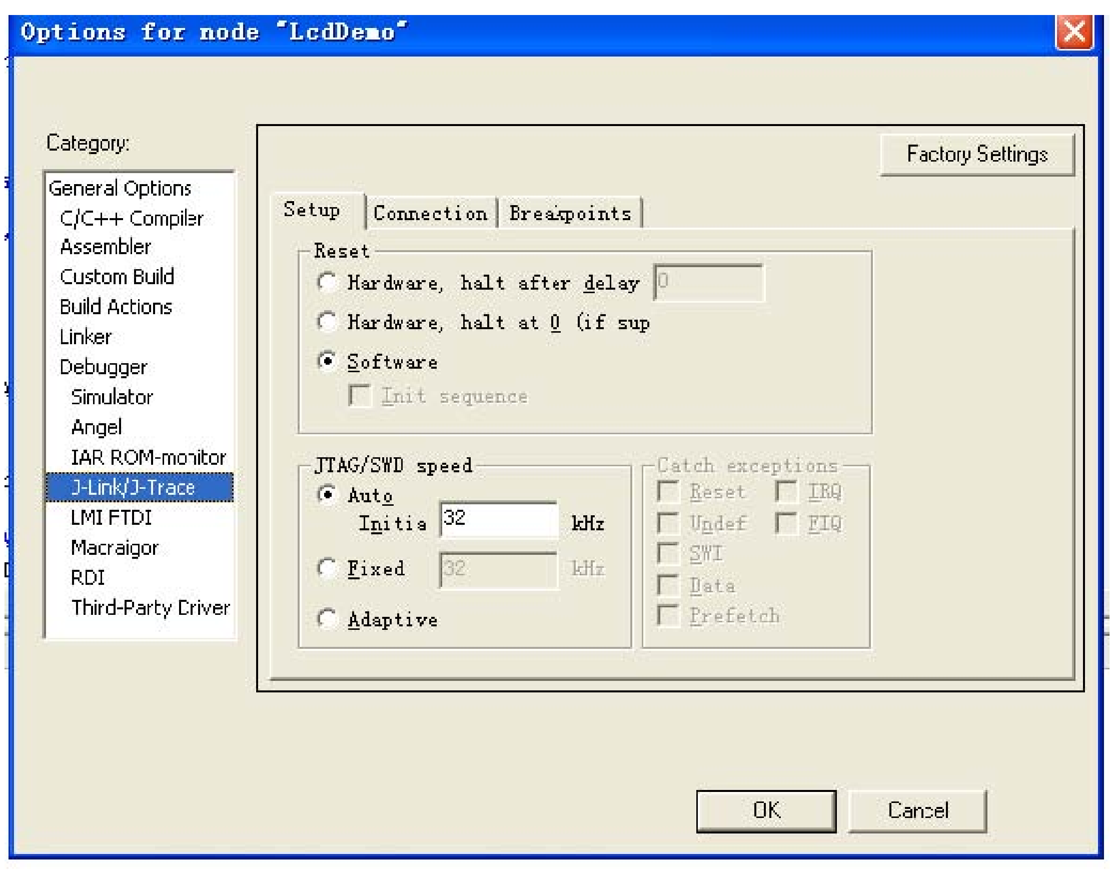


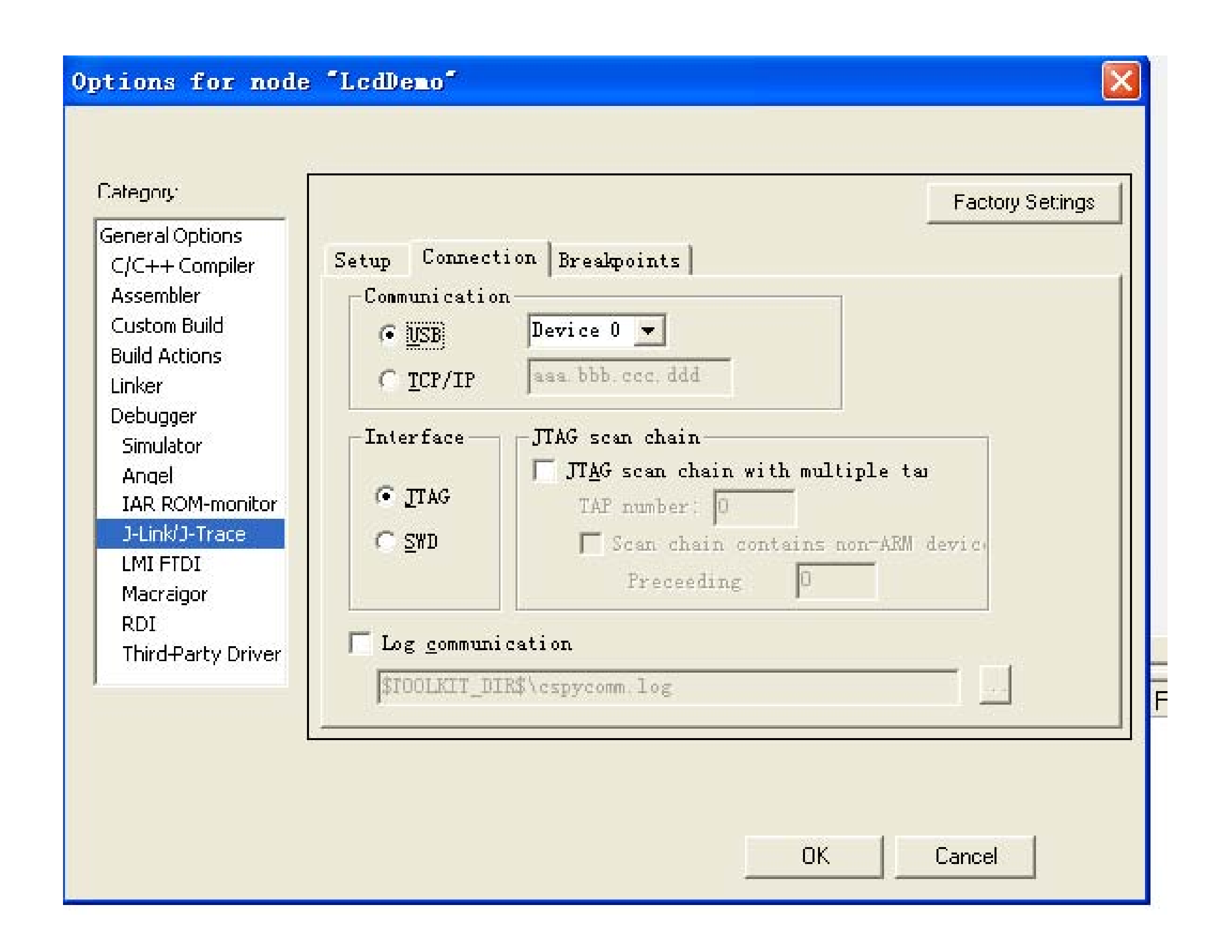
Options for node "LcdDemo"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Simulator
Angel
IAR ROM-monitor
J-Link/J-Trace
LMI FTDI
Macraigor
RDI
Third-Party Driver
Factory Settings
Setup
Download
Extra Options
Plugins
Driver
J-Link/J-Trace
Run to
main
Setup macros
Use macro file
Device description file
Override default
$TOOLKIT_DIR$\CONFIG\iolm3s101.ddf
OK
Cancel

选择 JLINK

设置：


Options for node "LcdDemo"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Simulator
Angel
IAR ROM-monitor
J-Link/J-Trace
LMI FTDI
Macraigor
RDI
Third-Party Driver
Factory Settings
Setup
Download
Extra Options
Plugins
Attach to progr
Verify download
Suppress downloa
Use flash loade
,,,, (default),
Edit...
OK
Cancel


Options for node "LcdDemo"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Simulator
Angel
IAR ROM-monitor
J-Link/J-Trace
LMI FTDI
Macraigor
RDI
Third-Party Driver
Factory Settings
Setup
Connection
Breakpoints
Reset
Hardware, halt after delay
0
Hardware, halt at 0 (if sup
Software
Init sequence
JTAG/SWD speed
Auto
Initia
32
kHz
Fixed
32
kHz
Adaptive
Catch exceptions
Reset
IRQ
Undef
FIQ
SWI
Data
Prefetch
OK
Cancel

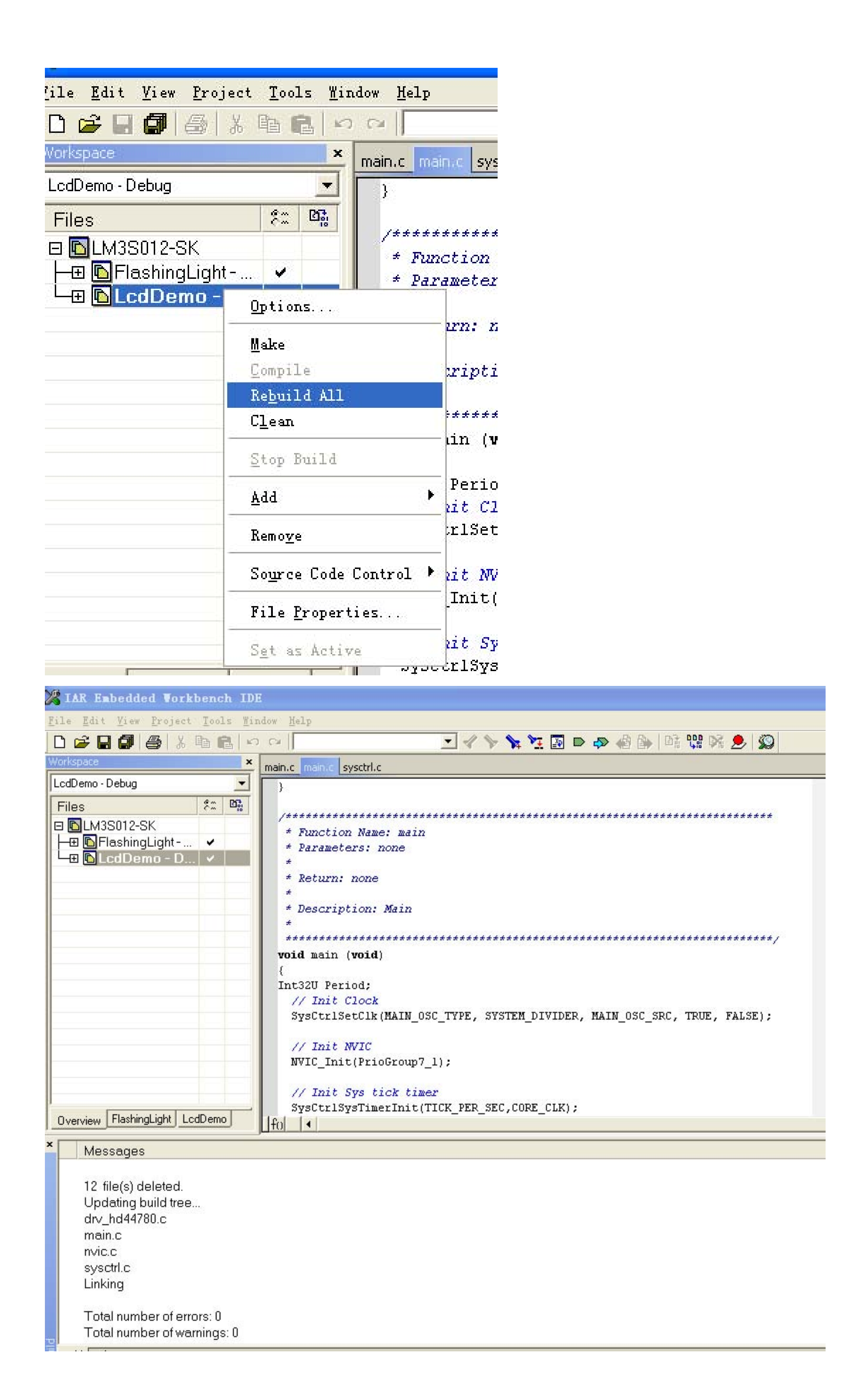


到这里基本完成设置了。。。

其实上面的设置，IAR 下这个例程已经默认设置好的了。

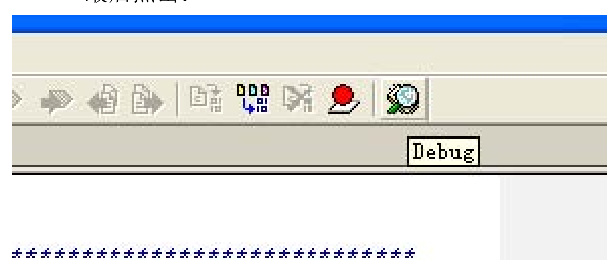
直接用就行了。

重新



File Edit View Project Tools Window Help
Workspace
LcdDemo - Debug
Files
LM3S012-SK
FlashingLight - ...
LcdDemo -
Options...
Make
Compile
Rebuild All
Clean
Stop Build
Add
Remove
Source Code Control
File Properties...
Set as Active
IAR Embedded Workbench IDE
main.c
sysctrl.c
/**
* Function Name: main
* Parameters: none
*
* Return: none
*
* Description: Main
*
**/
void main (void)
{
Int32U Period;
// Init Clock
SysCtrlSetClk(MAIN_OSC_TYPE, SYSTEM_DIVIDER, MAIN_OSC_SRC, TRUE, FALSE);
// Init NVIC
NVIC_Init(PrioGroup7_1);
// Init Sys tick timer
SysCtrlSysTimerInit(TICK_PER_SEC,CORE_CLK);
Overview
FlashingLight
LcdDemo
Messages
12 file(s) deleted.
Updating build tree...
drv_hd44780.c
main.c
nvic.c
sysctrl.c
Linking
Total number of errors: 0
Total number of warnings: 0

最后点击：



IAR Embedded Workbench IDE

File Edit View Project Debug Disassembly J-Link Tools Window Help

Workspace

LcdDemo - Debug

Files

LM3S012-SK
FlashingLi...
LcdDem...

Overview FlashingLight

main.c main.c sysctrl.c

```
}

/*************************************************************************
 * Function Name: main
 * Parameters: none
 *
 * Return: none
 *
 * Description: Main
 *
 *************************************************************************/
void main (void)
{
Int32U Period;
  // Init Clock
  SysCtrlSetClk(MAIN_OSC_TYPE, SYSTEM_DIVIDER, MAIN_OSC_SRC, TRUE, FALSE);

  // Init NVIC
```

Log

```
Sat Sep 15 12:10:59 2007: Loaded macro file: D:\Program Files\IAR Systems\Embedded Workbench 4.0 Kickstart\arm\config\flashloader\Luminary\FlashLM3Sxxx2k.mac
Sat Sep 15 12:10:59 2007: DLL version: V3.74i, compiled Sep 11 2007 14:43:07
Sat Sep 15 12:10:59 2007: Firmware: J-Link compiled Jun 28 2007 10:45:08 ARM Rev.5
Sat Sep 15 12:10:59 2007: JTAG speed is initially set to: 32 kHz
Sat Sep 15 12:10:59 2007: Found Cortex-M3, Little endian.
Sat Sep 15 12:10:59 2007: TPIU fitted.
Sat Sep 15 12:10:59 2007:  FPUnit: 6 code (BP) slots and 2 literal slots
Sat Sep 15 12:10:59 2007: Software reset was performed
Sat Sep 15 12:10:59 2007: Initial reset was performed
Sat Sep 15 12:10:59 2007: J-Link found 1 JTAG device. ARM core Id: 2BA00477(Cortex M3)
Sat Sep 15 12:10:59 2007: Device at TAP0 selected
Sat Sep 15 12:11:00 2007: Turning off watchdog
Sat Sep 15 12:11:00 2007: 1340 bytes downloaded and verified (2.33 Kbytes/sec)
```

Overview FlashingLight

Log

```
Sat Sep 15 12:10:59 2007: Firmware: J-Link compiled Jun 28 2007 10:45:08 ARM Rev.5
Sat Sep 15 12:10:59 2007: JTAG speed is initially set to: 32 kHz
Sat Sep 15 12:10:59 2007: Found Cortex-M3, Little endian.
Sat Sep 15 12:10:59 2007: TPIU fitted.
Sat Sep 15 12:10:59 2007:  FPUnit: 6 code (BP) slots and 2 literal slots
Sat Sep 15 12:10:59 2007: Software reset was performed
Sat Sep 15 12:10:59 2007: Initial reset was performed
Sat Sep 15 12:10:59 2007: J-Link found 1 JTAG device. ARM core Id: 2BA00477(Cortex M3)
Sat Sep 15 12:10:59 2007: Device at TAP0 selected
Sat Sep 15 12:11:00 2007: Turning off watchdog
Sat Sep 15 12:11:00 2007: 1340 bytes downloaded and verified (2.33 Kbytes/sec)
Sat Sep 15 12:11:00 2007: Loaded debugee: D:\Program Files\IAR Systems\Embedded Workbench 4.0 Kickstart\arm\cc
Sat Sep 15 12:11:00 2007: Found Cortex-M3, Little endian.
Sat Sep 15 12:11:00 2007: TPIU fitted.
Sat Sep 15 12:11:00 2007:  FPUnit: 6 code (BP) slots and 2 literal slots
Sat Sep 15 12:11:00 2007: Software reset was performed
Sat Sep 15 12:11:00 2007: Target reset
Sat Sep 15 12:11:02 2007: Program exit reached.
Sat Sep 15 12:11:02 2007: 3198 bytes downloaded into FLASH and verified (1.40 Kbytes/sec)
```

可以看到调试的一下 LOG 信息。

已经成功调试 Cortex-M3 内核芯片。

9. 附录四、J-Link各版本的特点和SWD 使用说明

V6 和 V7 版本的 J-Link 在硬件电路和软件方面都做了加强。硬件电路方面增加了 USB 保护，降低了仿真器功耗，拓展了接口电平支持范围；

软件方面更有重大改进，主要是支持了最新的 SWD 接口及 SWV (Serial Wire Viewer)，SWD 是 ARM 公司新推出的一种调试接口，它仅需要 2 条线即可进行调试，与传统的 4 线 JTAG 相比可以有效减少调试占用的口线资源，有效提高少引脚芯片的口线利用率。目前 SWD 接口主要存在 ortex-M3 内核的芯片上，如 ST 公司的 STM32 系列、Luminary 公司的 LM3S 系列。

注意，只有 V6 及以后版本的 J-Link 才支持 SWD！

V7.0 版本 J-LINK，除拥有上一版本 V6.0 的全部功能外，对于 Cortex-M3 的 Serial Wire Viewer(SWV)速度是 V6.0 的 12 倍

V8.0 版本相对 V7.0 版本的改进：

（1）SWD 硬件接口支持 1.2-5.0V 的目标板，V7.0 只能支持 3.3V 的目标板。

（2）V8.0 使用双色 LED 可以指示更多的工作状态，V7.0 只有 1 个 LED 指示灯。

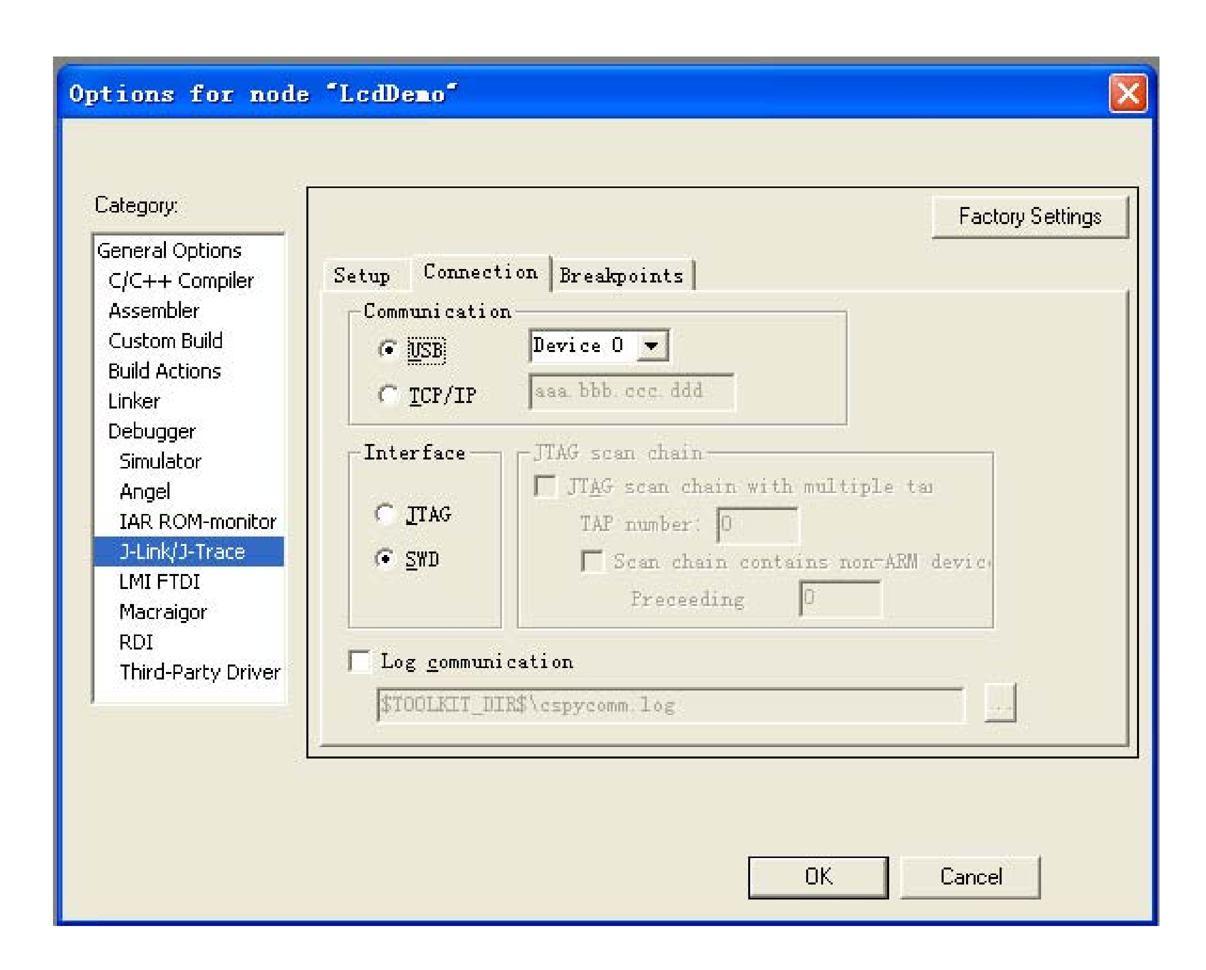
（3）V8.0 将 JTAG 接口限流电阻由 220 欧姆修改为 110 欧姆，增强了 JTAG 驱动能力，提高了目标板的兼容性。

（4）优化了固件结构，将固件升级功能移到 bootloader 区，使应用程序区扩大一倍，便于增加新的功能。

目前 SWD 只有 4.42 版本 IAR 才开始支持，SWD 设置很简单，如下图所示：

SWD 调试设置(JLINK V6.0 版本才支持)：

点击 Lcddemo 右键 option—>j-link/j-trace->connection 中设置



设置好后，设 OK 退出设置

点击调试：



进入调试后：

Tue Dec 25 11:11:04 2007: Loaded macro file: D:\Program Files\IAR Systems\Embedded Workbench 4.0 Evaluation\ARM\config\flashloader\Luminary\FlashLM3Sxxx2k.mac

Tue Dec 25 11:11:04 2007: DLL version: V3.78c, compiled Dec 13 2007 20:28:39

Tue Dec 25 11:11:04 2007: Firmware: J-Link ARM V6 compiled Dec 03 2007 17:34:18

Tue Dec 25 11:11:04 2007: Selecting SWD as current target interface.

Tue Dec 25 11:11:04 2007: JTAG speed is initially set to: 32 kHz

Tue Dec 25 11:11:04 2007: Found SWD-DP with ID 0ba01477

Tue Dec 25 11:11:05 2007: TPIU fitted.

Tue Dec 25 11:11:05 2007:   FPUnit: 6 code (BP) slots and 2 literal slots

Tue Dec 25 11:11:05 2007: Found SWD-DP with ID 0ba01477

Tue Dec 25 11:11:05 2007: TPIU fitted.

Tue Dec 25 11:11:05 2007:   FPUnit: 6 code (BP) slots and 2 literal slots

Tue Dec 25 11:11:05 2007: Software reset was performed

Tue Dec 25 11:11:05 2007: Initial reset was performed

Tue Dec 25 11:11:05 2007: Turning off watchdog

Tue Dec 25 11:11:06 2007: 1340 bytes downloaded and verified (1.71 Kbytes/sec)

Tue Dec 25 11:11:06 2007: Loaded debugee: D:\Program Files\IAR Systems\Embedded Workbench 4.0

Evaluation\ARM\config\flashloader\Luminary\FlashLM3Sxxx2k.d79

Tue Dec 25 11:11:06 2007: Found SWD-DP with ID 0ba01477

Tue Dec 25 11:11:06 2007: TPIU fitted.

Tue Dec 25 11:11:06 2007: FPUnit: 6 code (BP) slots and 2 literal slots

Tue Dec 25 11:11:06 2007: Software reset was performed

Tue Dec 25 11:11:06 2007: Target reset

Tue Dec 25 11:11:12 2007: Program exit reached.

Tue Dec 25 11:11:13 2007: 3198 bytes downloaded into FLASH and verified (0.47 Kbytes/sec)

Tue Dec 25 11:11:13 2007: Loaded debugee: D:\Program Files\IAR Systems\Embedded Workbench 4.0 Evaluation\ARM\examples\Luminary\LM3S102-SK\LCD\Debug\Exe\LcdDemo.d79

Tue Dec 25 11:11:13 2007: Target reset

```
Log
Tue Dec 25 11:11:04 2007: Loaded macro file: D:\Program Files\IAR Systems\Embedded Workbench 4.0 Evaluation\ARM\config\flashloader\Luminary\FlashLM3Sxxx2k.mac
Tue Dec 25 11:11:04 2007: DLL version: V3.78c, compiled Dec 13 2007 20:28:39
Tue Dec 25 11:11:04 2007: Firmware: J-Link ARM V6 compiled Dec 03 2007 17:34:18
Tue Dec 25 11:11:04 2007: Selecting SWD as current target interface.
Tue Dec 25 11:11:04 2007: JTAG speed is initially set to: 32 kHz
Tue Dec 25 11:11:04 2007: Found SWD-DP with ID 0ba01477
Tue Dec 25 11:11:05 2007: TPIU fitted.
Tue Dec 25 11:11:05 2007:   FPUnit: 6 code (BP) slots and 2 literal slots
Tue Dec 25 11:11:05 2007: Found SWD-DP with ID 0ba01477
Tue Dec 25 11:11:05 2007: TPIU fitted.
Tue Dec 25 11:11:05 2007:   FPUnit: 6 code (BP) slots and 2 literal slots
Tue Dec 25 11:11:05 2007: Software reset was performed
Tue Dec 25 11:11:05 2007: Initial reset was performed
Tue Dec 25 11:11:05 2007: Turning off watchdog
```

IAR Embedded Workbench IDE

File Edit View Project Debug Disassembly J-Link Tools Window Help

Workspace: LcdDemo - Debug — Files: LM3S012-SK, FlashingLi..., LcdDem... — Overview | FlashingLight | L

main.c

```
void main (void)
{
Int32U Period;
  // Init Clock
  SysCtrlSetClk(MAIN_OSC_TYPE, SYSTEM_DIVIDER, MAIN_OSC_SRC, TRUE, FALSE);

  // Init NVIC
  NVIC_Init(PrioGroup7_1);

  // Init Sys tick timer
  SysCtrlSysTimerInit(TICK_PER_SEC,CORE_CLK);
  NVIC_IntrRegister(SysTickIntrHandler,STI,0x40);
  NVIC_IntrEnable(STI);
  SysCtrlSysTimerEna();

  // IO Inittialize
  RCGC2_bit.PORTB = 1;          // PORTB clock enable
```


```
Log
Tue Dec 25 11:11:04 2007: Loaded macro file: D:\Program Files\IAR Systems\Embedded Workbench 4.0 Evaluation\ARM\config
Tue Dec 25 11:11:04 2007: DLL version: V3.78c, compiled Dec 13 2007 20:28:39
Tue Dec 25 11:11:04 2007: Firmware: J-Link ARM V6 compiled Dec 03 2007 17:34:18
Tue Dec 25 11:11:04 2007: Selecting SWD as current target interface.
Tue Dec 25 11:11:04 2007: JTAG speed is initially set to: 32 kHz
Tue Dec 25 11:11:04 2007: Found SWD-DP with ID 0ba01477
Tue Dec 25 11:11:05 2007: TPIU fitted.
Tue Dec 25 11:11:05 2007:   FPUnit: 6 code (BP) slots and 2 literal slots
Tue Dec 25 11:11:05 2007: Found SWD-DP with ID 0ba01477
Tue Dec 25 11:11:05 2007: TPIU fitted.
Tue Dec 25 11:11:05 2007:   FPUnit: 6 code (BP) slots and 2 literal slots
Tue Dec 25 11:11:05 2007: Software reset was performed
Tue Dec 25 11:11:05 2007: Initial reset was performed
Tue Dec 25 11:11:05 2007: Turning off watchdog
```

10. 附 录 五 、 MDK-ARM 3.2 下 J-LINK/J-TRACE使用说明

背景：

MDK3.2 以上版本已经支持 J-link/J-trace 的调试

具体出自：www.keil.com

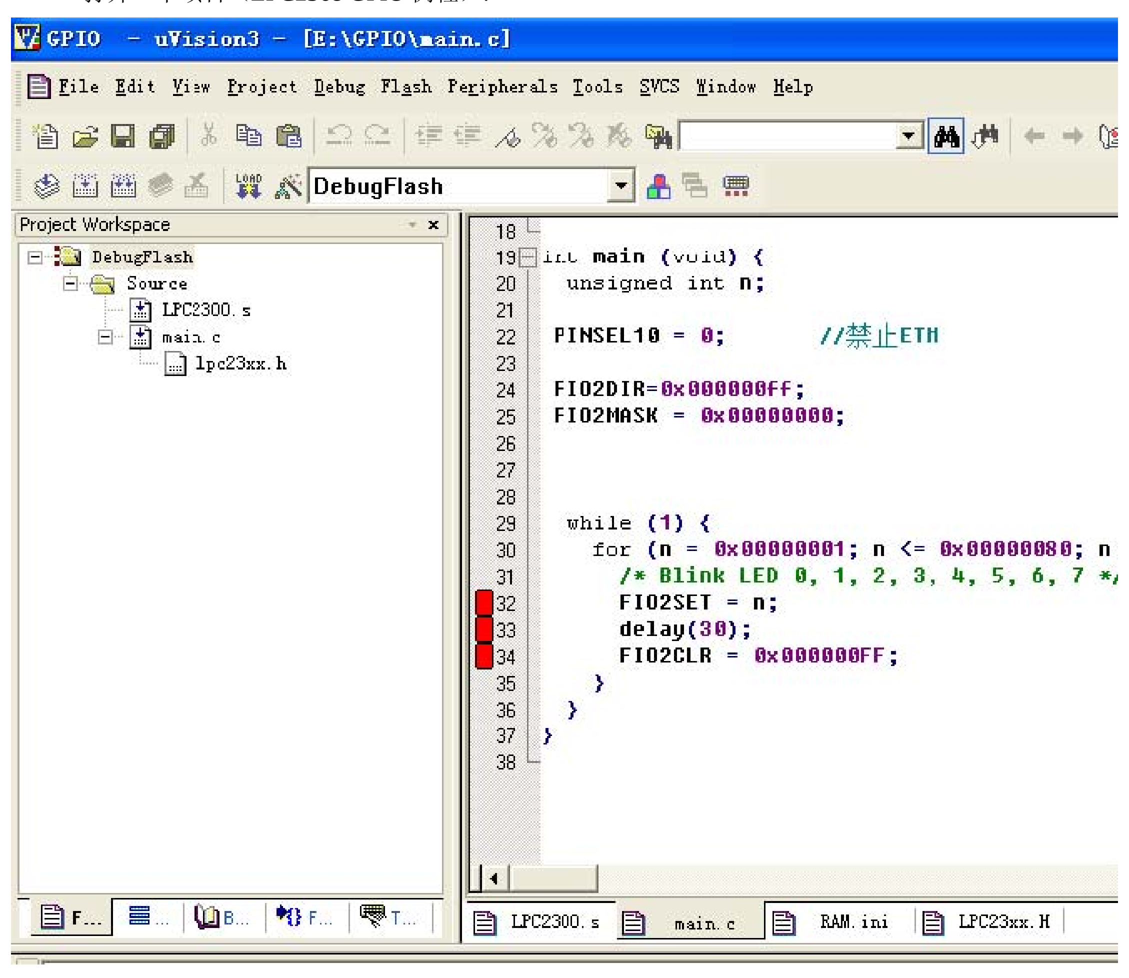
RealView MDK support for SEGGER J-Link and J-Trace

http://www.keil.com/pr/article/1141.htm

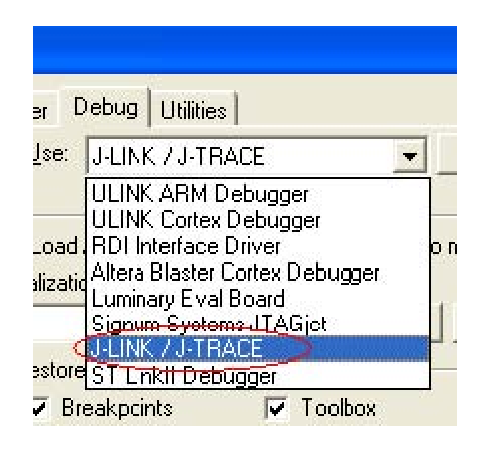
MDK3.2 以上版本用 J-link/J-trace 来调试的一些设置(非 RDI 方式).

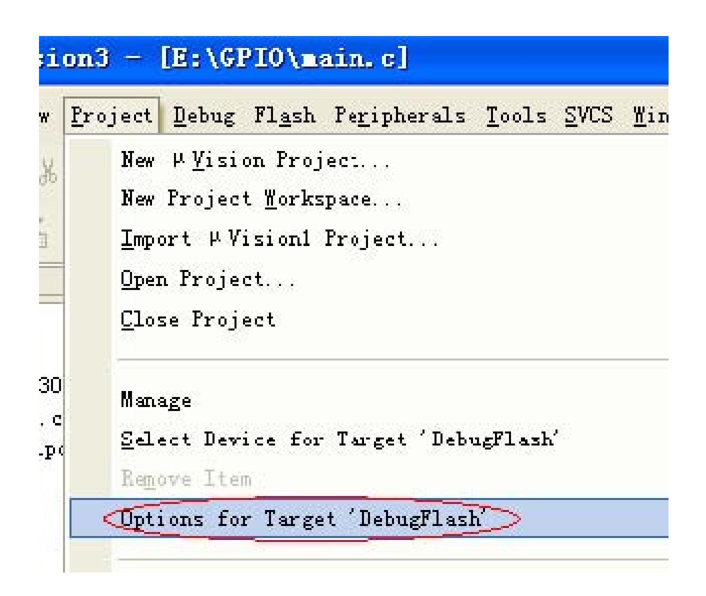
这种方式将会支持 ARM7/ARM9/Cortex-M3 等内核的调试.

打开一个项目（LPC2368 GPIO 例程）:

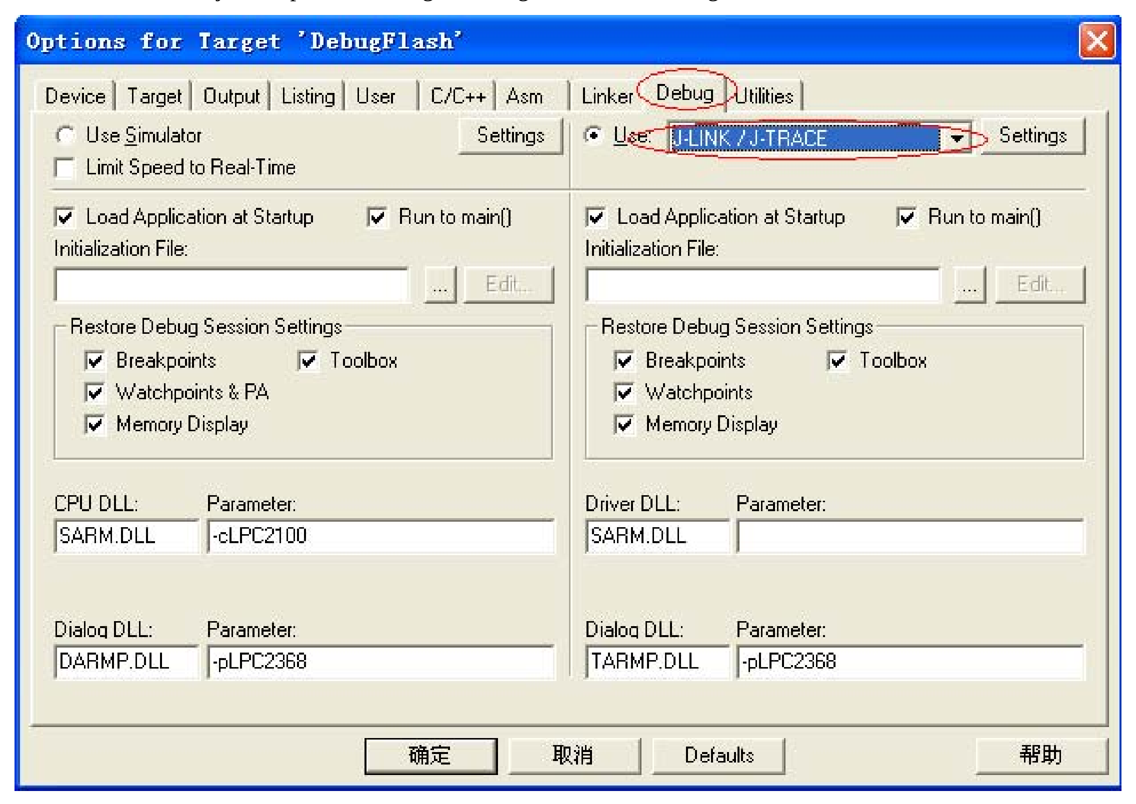


菜单 Project->Options for Target ‘DebugFlash’

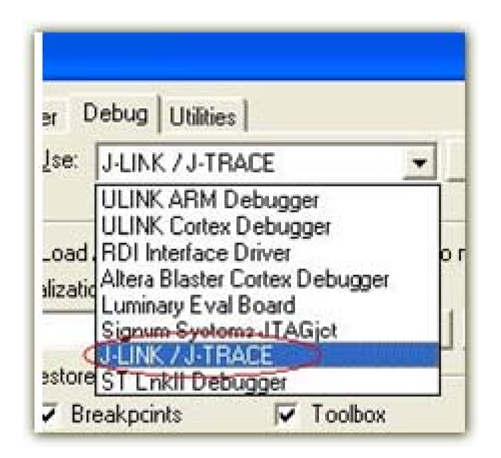




2．进入: Project->Options for Target ‘DebugFlash’ 点击 Debug 选项卡



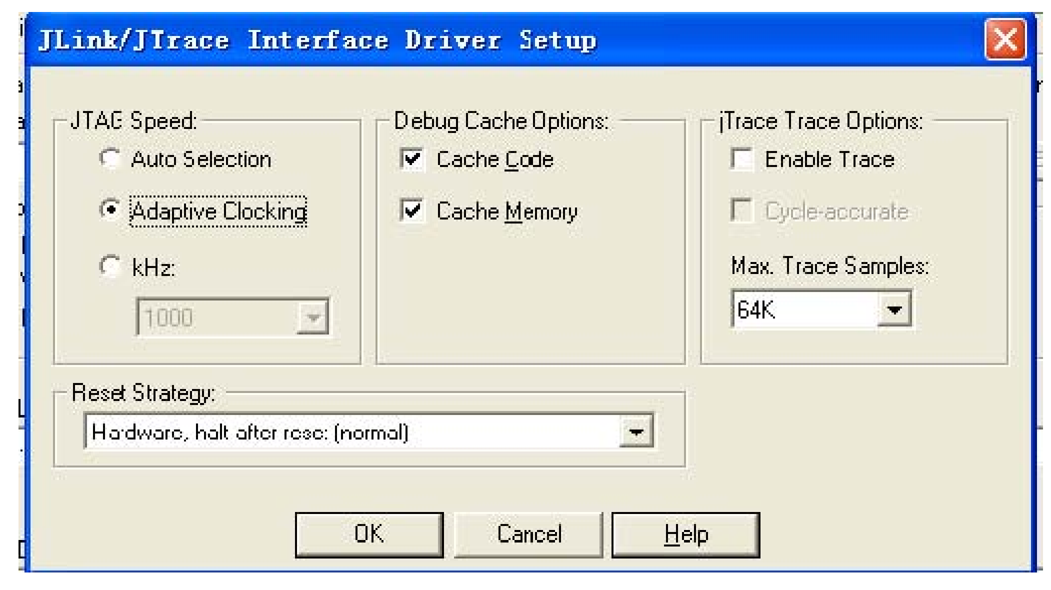
选择 J-LINK/J-TRACE



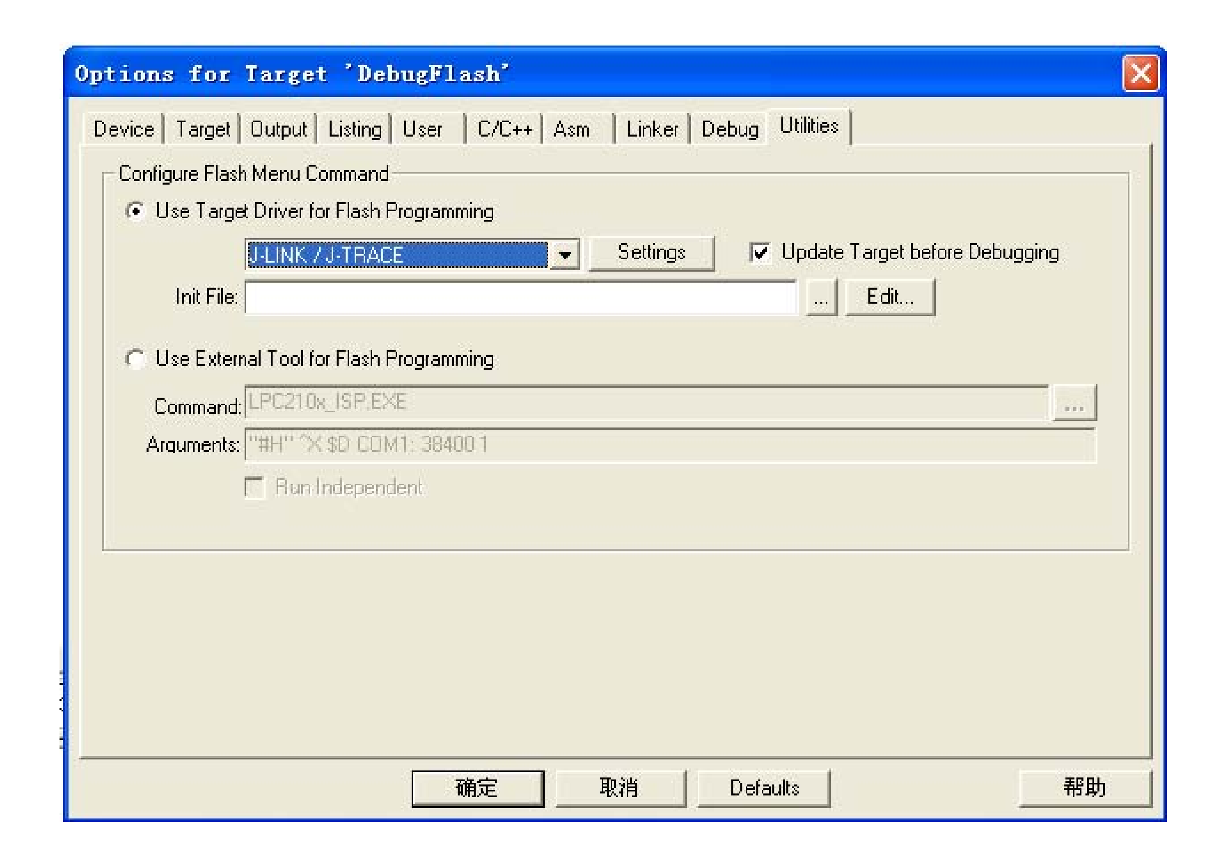
Debug->settings



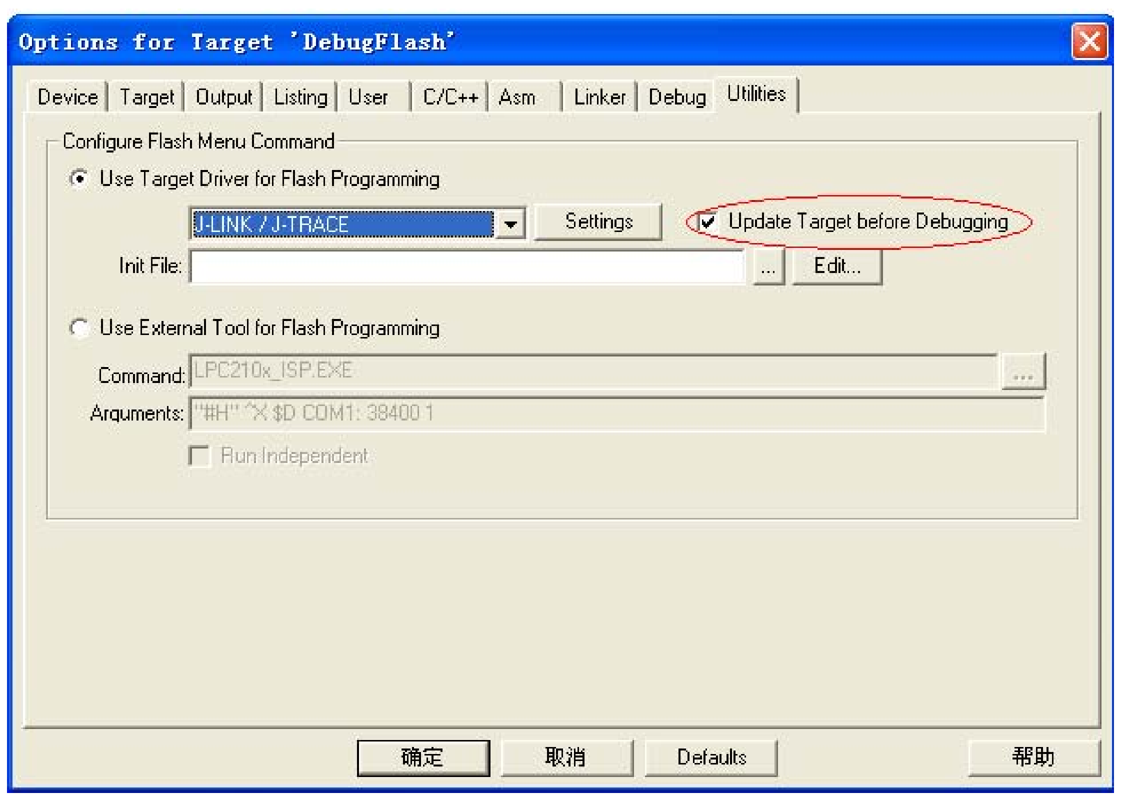
4．J-LINK/J-TRACE Interface Driver Setup 中按默认设置即可,如 NXP 系列的可以选取 Adaptive Clocking (这里是按 NXP LPC2368 为例的)



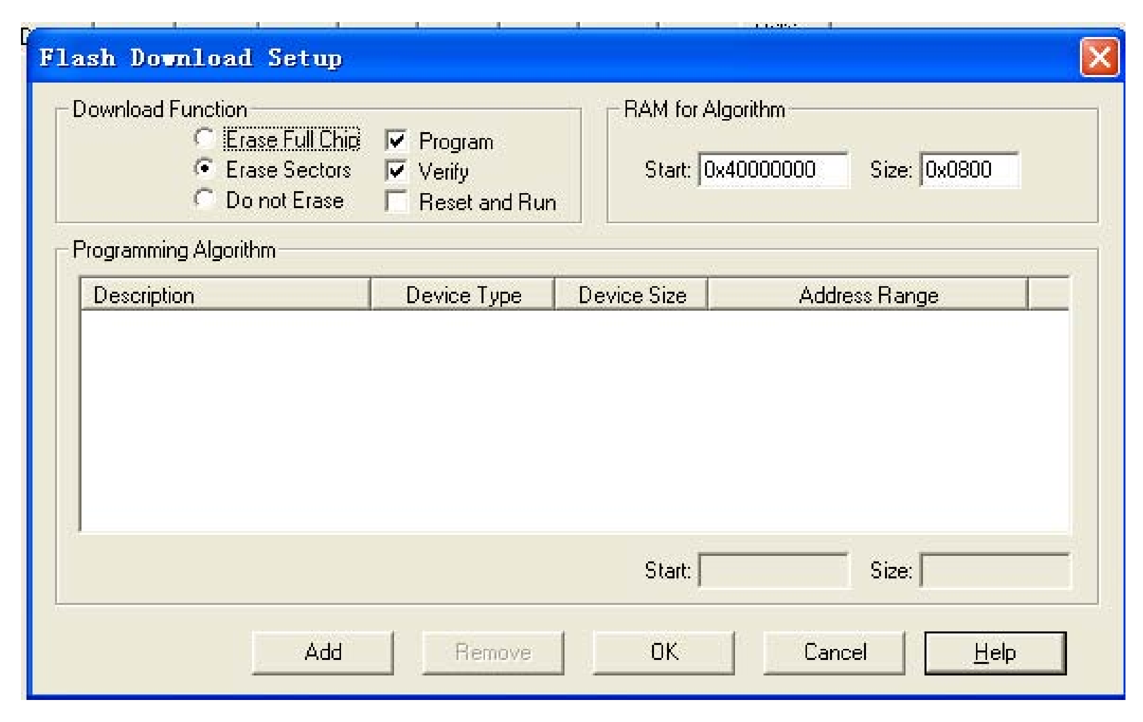
点 OK 退出.然后在 Utilities 选项中设置 FLASH 算法.



5．Update Target before Debugging 此项打勾,只要你的程序有更新,在进入调试前会自动选烧录FLASH.

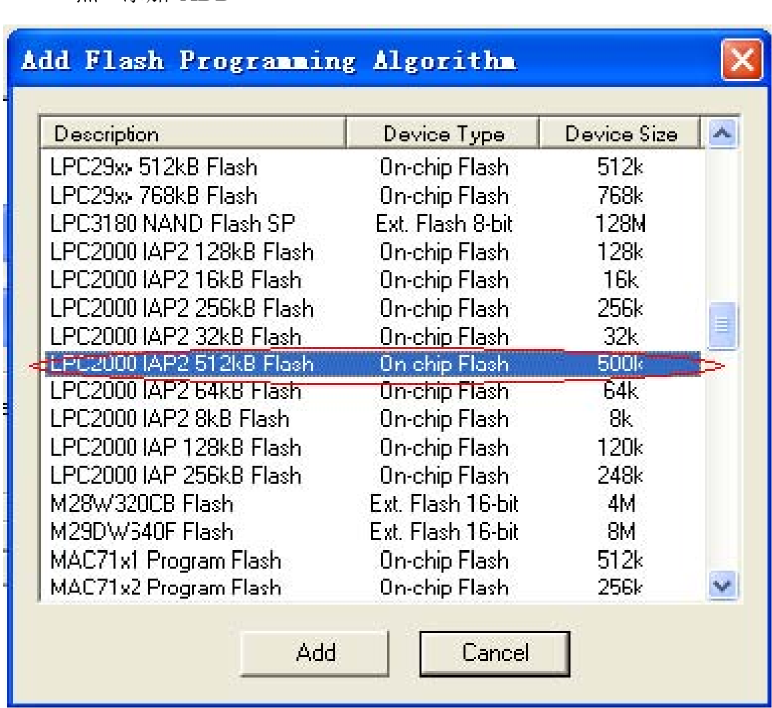


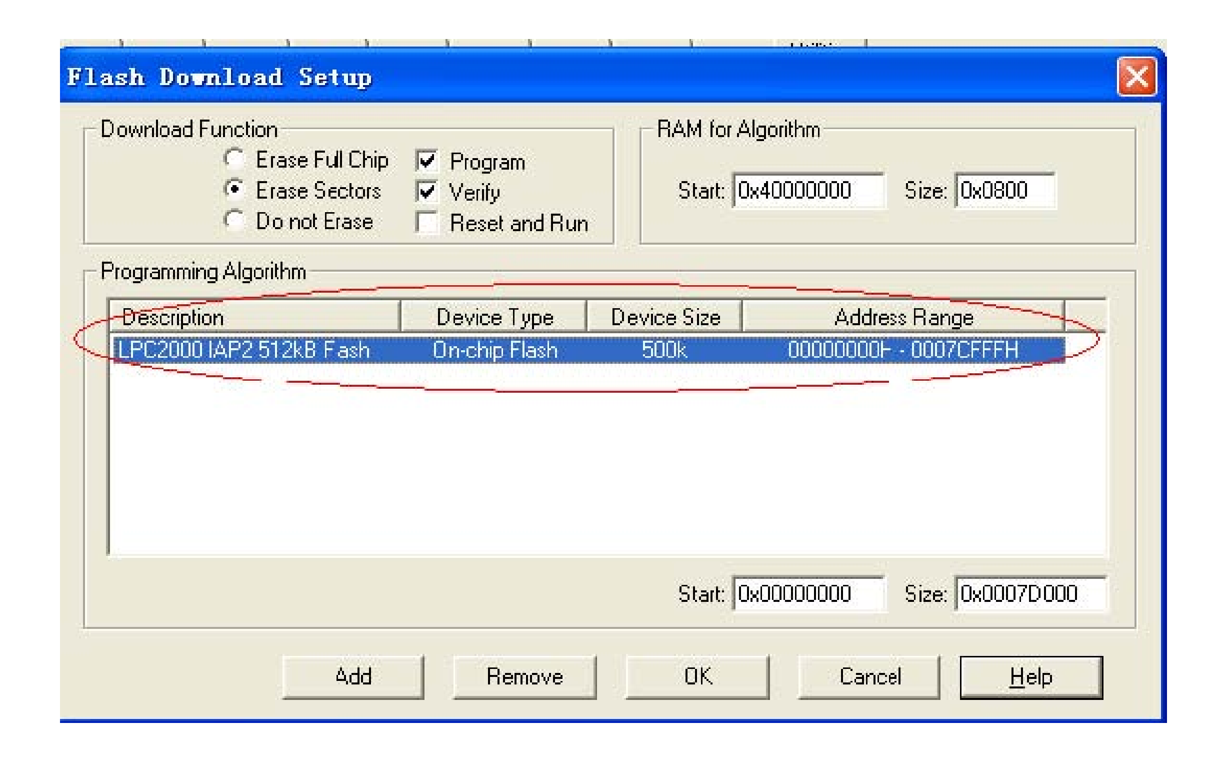
点 settings



由于我的目标芯片是 LPC2368,其 FLASH 大小为 500K

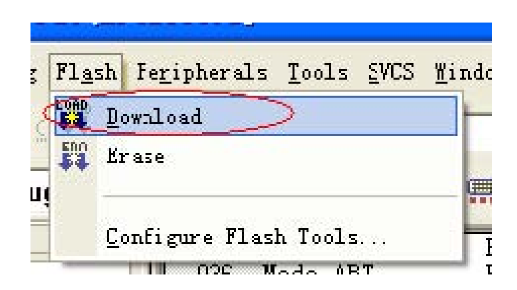
点”添加”ADD

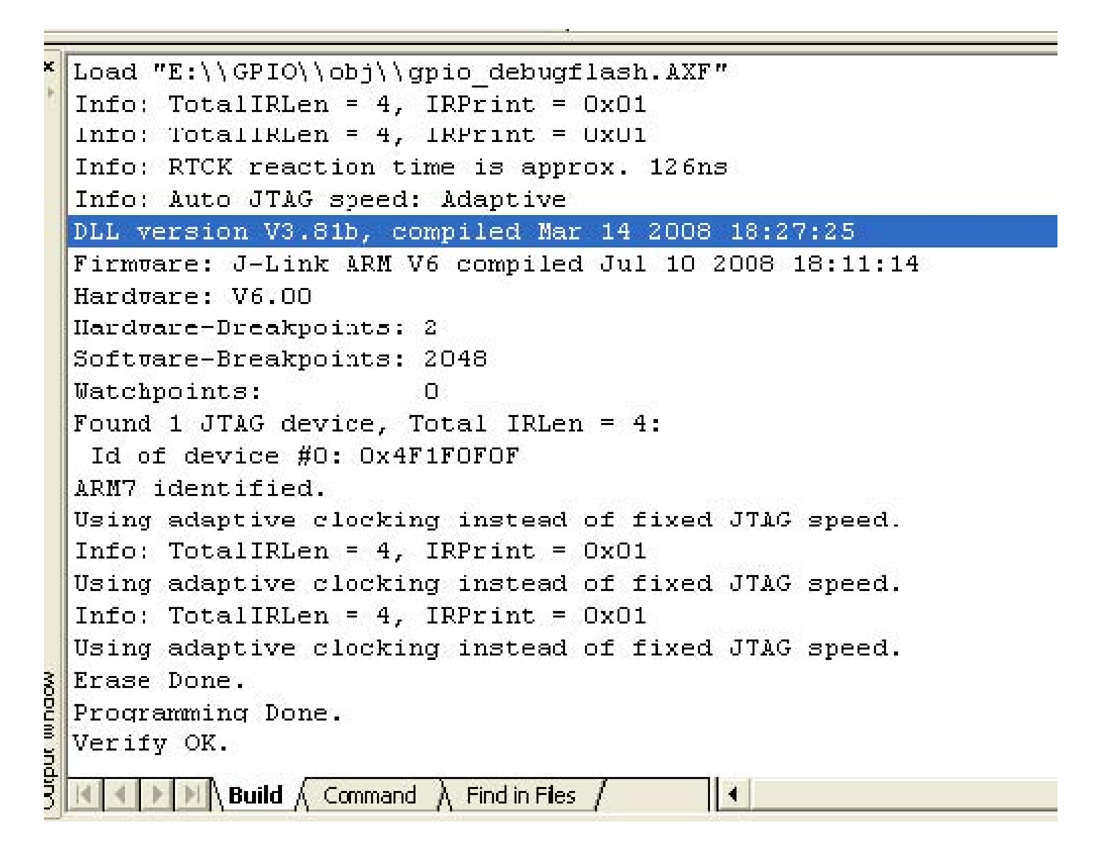




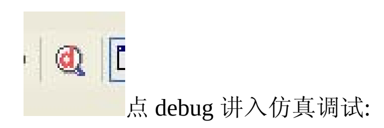
点击 OK 退出设置.

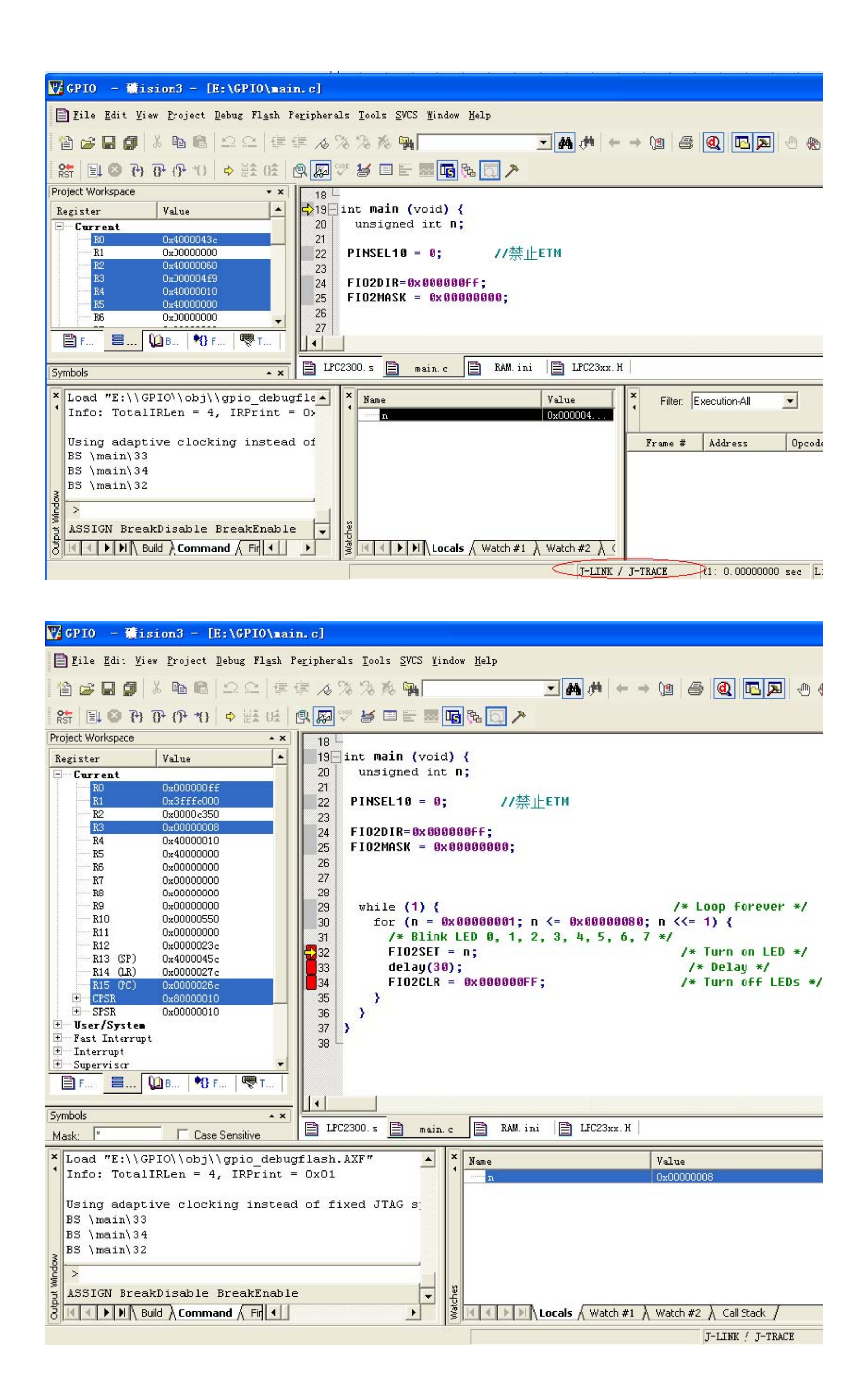
6．菜单 Flash ->Download（这里是试验烧写效果）





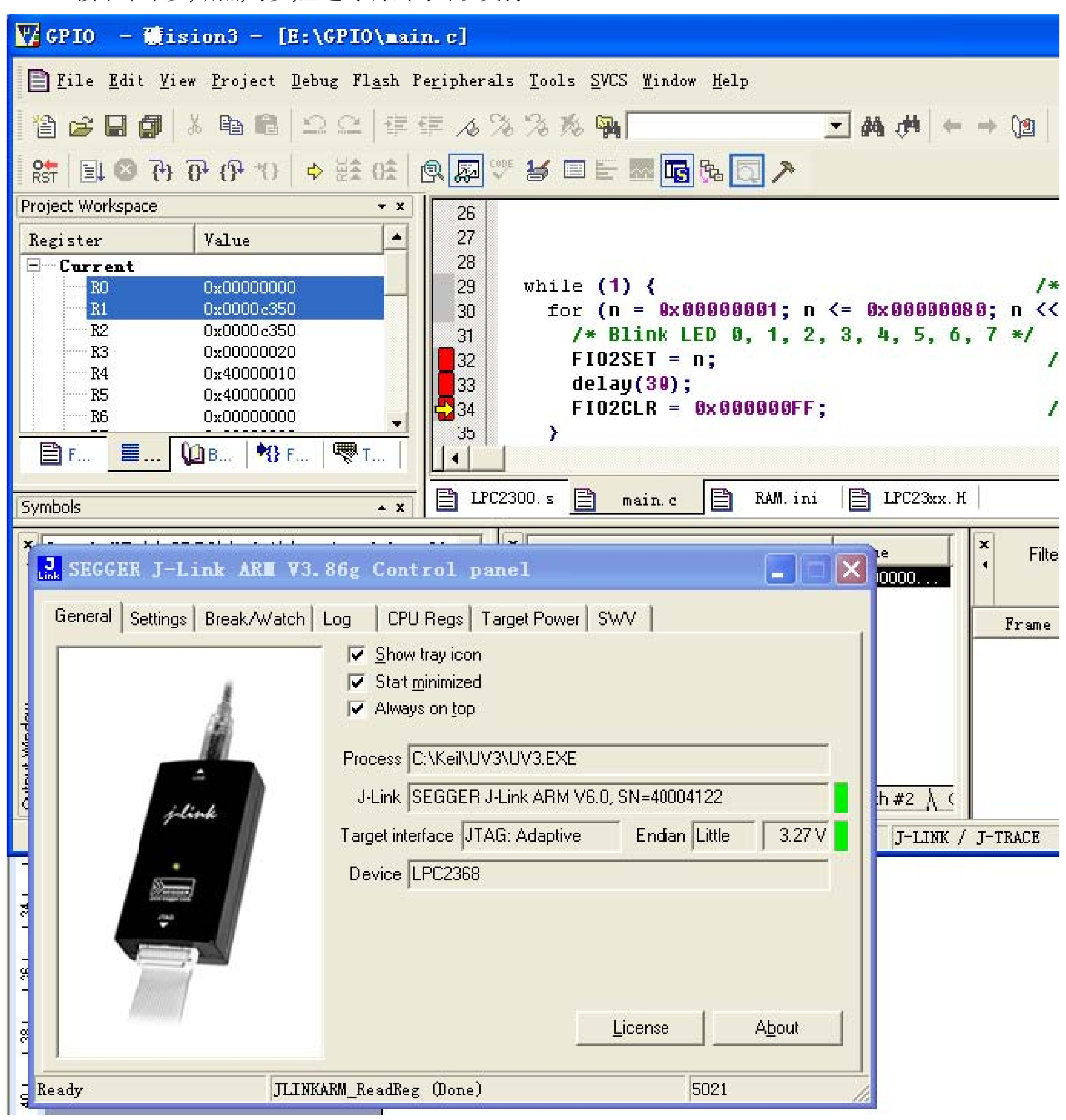
可以看到芯片擦除,编程,校验 OK.

点 debug 讲入仿真调试:



GPIO - µVision3 - [E:\GPIO\main.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register Value
Current
R0 0x4000043c
R1 0x00000000
R2 0x40000060
R3 0x000004f9
R4 0x40000010
R5 0x40000000
R6 0x00000000
int main (void) {
unsigned int n;
PINSEL10 = 0; //禁止ETM
FIO2DIR=0x000000ff;
FIO2MASK = 0x00000000;
LPC2300.s main.c RAM.ini LPC23xx.H
Symbols
Load "E:\\GPIO\\obj\\gpio_debugfla
Info: TotalIRLen = 4, IRPrint = 0x
Using adaptive clocking instead of
BS \main\33
BS \main\34
BS \main\32
ASSIGN BreakDisable BreakEnable
Build Command
Name Value
n 0x000004...
Locals Watch #1 Watch #2
Filter: Execution-All
Frame # Address Opcode
J-LINK / J-TRACE
t1: 0.00000000 sec
GPIO - µVision3 - [E:\GPIO\main.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register Value
Current
R0 0x000000ff
R1 0x3fffc000
R2 0x0000c350
R3 0x00000008
R4 0x40000010
R5 0x40000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000550
R11 0x00000000
R12 0x0000023c
R13 (SP) 0x4000045c
R14 (LR) 0x0000027c
R15 (PC) 0x0000026c
CPSR 0x80000010
SPSR 0x00000010
User/System
Fast Interrupt
Interrupt
Supervisor
int main (void) {
unsigned int n;
PINSEL10 = 0; //禁止ETM
FIO2DIR=0x000000FF;
FIO2MASK = 0x00000000;
while (1) { /* Loop forever */
for (n = 0x00000001; n <= 0x00000080; n <<= 1) {
/* Blink LED 0, 1, 2, 3, 4, 5, 6, 7 */
FIO2SET = n; /* Turn on LED */
delay(30); /* Delay */
FIO2CLR = 0x000000FF; /* Turn off LEDs */
}
}
}
Symbols
Mask: * Case Sensitive
LPC2300.s main.c RAM.ini LPC23xx.H
Load "E:\\GPIO\\obj\\gpio_debugflash.AXF"
Info: TotalIRLen = 4, IRPrint = 0x01
Using adaptive clocking instead of fixed JTAG s
BS \main\33
BS \main\34
BS \main\32
ASSIGN BreakDisable BreakEnable
Build Command
Name Value
n 0x00000008
Locals Watch #1 Watch #2 Call Stack
J-LINK / J-TRACE

接下来单步,断点,跨步,全速等调试均可以支持.



在 KEIL 下用 J-LINK/J-TRACE 效果还是不错的。